THE UNIVERSITY *of* EDINBURGH

Verification Problems for Timed and Probabilistic Extensions of Petri Nets

Radu Ciobanu

Doctor of Philosophy

Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh

2019

Abstract

In the first part of the thesis, we prove the decidability (and PSPACE-completeness) of the universal safety property on a timed extension of Petri Nets, called Timed Petri Nets. Every token has a real-valued clock (a.k.a. age), and transition firing is constrained by the clock values that have integer bounds (using strict and non-strict inequalities). The newly created tokens can either inherit the age from an input token of the transition or it can be reset to zero.

In the second part of the thesis, we refer to systems with controlled behaviour that are probabilistic extensions of VASS and One-Counter Automata. Firstly, we consider infinite state Markov Decision Processes (MDPs) that are induced by probabilistic extensions of VASS, called VASS-MDPs. We show that most of the qualitative problems for general VASS-MDPs are undecidable, and consider a monotone subclass in which only the controller can change the counter values, called 1-VASS-MDPs. In particular, we show that limit-sure control state reachability for 1-VASS-MDPs is decidable, i.e., checking whether one can reach a set of control states with probability arbitrarily close to 1. Unlike for finite state MDPs, the control state reachability property may hold limit surely (i.e. using an infinite family of strategies, each of which achieving the objective with probability $\geq 1-\varepsilon$, for every $\varepsilon > 0$), but not almost surely (i.e. with probability 1). Secondly, we consider infinite state MDPs that are induced by probabilistic extensions of One-Counter Automata, called One-Counter Markov Decision Processes (OC-MDPs). We show that the almost-sure $\{1,2,3\}$-Parity problem for OC-MDPs is at least as hard as the limit-sure selective termination problem for OC-MDPs, in which one would like to reach a particular set of control states and counter value zero with probability arbitrarily close to 1.

Lay Summary

This thesis studies theoretical questions related to formal verification. Formal verification consists in proving (or disproving) the correctness of an algorithm underlying a system with respect to a certain mathematical property. In theoretical computer science, a system is a mathematical object which can take many forms. The algorithmic problems that we study in this thesis are related to two different kinds of systems: timed and probabilistic. Some of the most important systems that exist in the theoretical computer science literature are the Petri nets. These are a formal model for concurrent computation. What we study here is a more extended variant, called Timed Petri nets. We allow tokens to have an age (or clock value), and of course, these can become older when time passes. A safety property requires that nothing bad happens during an execution of the system. We establish the computational complexity of, and develop an algorithm for checking safety properties on Timed Petri nets.

We introduce a probabilistic extension of Petri nets, called VASS-MDPs, which are a model for concurrent computations with uncertainty. We show that, under certain restrictions (and unlike in the general case), there exists an algorithm for checking whether a target state can be reached with probability arbitrarily close to 1.

We also study properties on probabilistic one-counter systems, which are called One-counter Markov Decision Processes (OC-MDPs). These systems are similar to VASS-MDPs, with the mention that in this framework, there is only one counter, which can be tested for zero. We study two different computational problems and show a connection between almost-sure (a single strategy attains probability 1) and limit-sure (a family of strategies arbitrarily closely approximates 1) checking problems.

Acknowledgements

First of all, I would like to thank my supervisors Dr Richard Mayr and Dr Kousha Etessami for their continuous support and encouragement. They introduced me to a very interesting area of research, provided feedback on my progress and work, much of which is based on their earlier contributions. I gratefully acknowledge Dr Richard Mayr for guiding me through an unknown and rewarding road.

I am especially grateful to Patrick Totzke for numerous and very fruitful discussions and feedback, for long hours spent together tackling hard problems, for his wise advice, and for being a good friend.

I thank everyone at the LFCS for creating a very welcoming and friendly environment. In particular, I thank my fellow PhD student Ricardo Almeida for inviting me to social events, taught me about Portuguese culture and for being a good friend. *Obrigado*!

I thank Dr Parosh Abdulla and his research lab for having the chance to visit Uppsala and for many interesting and useful discussions.

I gratefully acknowledge financial support from EPSRC and LFCS who funded my stay in Edinburgh as well as conferences I attended.

I am very grateful to my family, especially to my parents and grandmother, for their love, support and for believing in me when I did not.

I am very grateful to Mihaela for her trust, her long wait and for being my person.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Radu Ciobanu*)

02.05.2019

Table of Contents

Chapter 1

Introduction

The subject of this thesis is to study algorithmic problems on certain classes of finitely representable systems whose underlying structure induce infinite state transition systems with timed or stochastic behaviour.

The motivation of this project comes from the need of developing mathematical models and techniques for verification problems that may occur in the outside world. Recall the case of verifying nuclear power management systems or aircraft sensors. For example, one would like to know whether a certain bad scenario might occur, and under which conditions. Doing an empirical analysis via trial-and-error could be time consuming, incomplete, and unreliable. Therefore, constructing a reliable system and reasoning about its behaviour is a very important topic in both theoretical and applied research. Property checking, also known as *model checking* [16], is based on verifying whether an abstract model of a system that is given as input meets a certain specification. Several tools have been developed to verify hardware properties, such as PRISM [62] and UPPAAL [18]. In order to perform this analysis algorithmically, one needs to define the problem in a clear mathematical language [1] .

In finite-state verification, a system is represented as a graph with a finite number of nodes which model the states of the program that we want to verify, whereas edges denote a transition relation between them. Under certain conditions, it may be the case that the transition system is infinite. Therefore, techniques from the finite-state systems framework may not be applicable. We recall the case of systems that have timing constraints [11], infinite state probabilistic systems, or systems with variables that range over infinite domains [63]. Hence, one would like to develop new verification techniques for verifying properties for an infinite-state framework.

[1] Note that some properties may not be verified algorithmically, due to undecidability constraints.

In the first part of the thesis, we study a decision problem on a timed extension of Petri nets framework, also known as Timed Petri Nets (TPNs). Every token has a real-valued clock (a.k.a. age), and transition firing is constrained by the clock values that have integer bounds (using strict and non-strict inequalities). The newly created tokens can either inherit the age from an input token of the transition or it can be reset to zero.

In the second part of the thesis, we refer to systems with both probabilistic and controlled behaviour. We study decision problems on *countably infinite* and *finitely branching* Markov Decision Processes (MDPs) that are derived from finitely represented transition systems, such as probabilistic extensions of Vector Addition Systems with States (VASS-MDPs [1]) and One-Counter Markov Decision Processes (OC-MDPs [28]).

The study of finite state systems with probabilistic behaviour is not new. Generally, these systems are modelled as Markov Chains (MCs), if their behaviour is purely probabilistic, or as Markov Decision Processes (MDPs) [76], if the systems exhibit both probabilistic and controlled behaviour.

In general, the mathematical techniques required to solve problems on finite state MDPs would not work in an infinite state environment, hence the need for new approaches [2]. A large number of finitely-representable infinite state MDPs has been already useful in modelling problems in various fields, such as queuing theory (Quasi-Birth-Death Processes) [41], [52], [17], model checking [16], natural language processing (Stochastic Context-Free Grammars) [56], systems biology [36], population dynamics and behaviour [86], etc.

All classes of systems that we are going to study in this project, such as Timed Petri Nets (TPNs), VASS-MDPs and OC-MDPs are related in the sense that they are finitely representable and their underlying structure induce infinite state transition systems. Note that VASS and Petri nets are mathematically equivalent [15]. However, they present certain unique characteristics. In OC-MDPs, each control state allows a test for the case where the counter value is zero, a property which is not valid for VASS-like systems. Therefore, every control state has two types of transitions, depending whether the counter value is strictly positive, or not. Also, in TPNs, there is no probabilistic behaviour, along with the fact that there exist two types of transitions, allowing for the time to pass and increase the age of tokens, and going from one state of the system to another, respectively.

[2] A simple example can be encountered in Section 1.4.4

1.1 Main Results

There are three main results that have been achieved during this project.

1. We show that the *existential Coverability* problem for TPNs is decidable and PSPACE-complete. The *Existential Coverability* problem asks, for a given place p and transition t, whether there exists a number m such that the marking $M(m) \stackrel{\text{def}}{=} m \cdot \{(p,0)\}$ ultimately enables t. Here, $M(m)$ contains exactly m tokens on place p with all clocks set to zero and *no other tokens*. This problem corresponds to checking safety properties in distributed networks of arbitrarily many (namely m) initially identical timed processes that communicate by handshake. A negative answer certifies that the 'bad event' of transition t can never happen regardless of the number m of processes, i.e., the network is safe for any size. Thus by checking existential coverability, one solves the dual problem of *Universal Safety*.

2. We study decidability questions on infinite state MDPs which are derived from a particular class of (finitely representable) probabilistic extension of Vector Addition Systems with States (VASS). This probabilistic extension of VASS is called *VASS-MDP* and has been introduced and studied in [1]. We show that a lot of qualitative problems are undecidable even for this variant. Hence, we focus on some particular monotone subclasses of VASS-MDPs, which are called *single sided*, depending whether if only the controller (1-VASS-MDPs) or the probabilistic player (P-VASS-MDPs) can change the counter values. We show that the *limit sure (control state) reachability* problem for 1-VASS-MDPs is *decidable*.

3. We study qualitative problems on infinite state MDPs which are derived from a probabilistic extension of One-Counter Automata, called One-Counter Markov Decision processes (OC-MDPs). We show that in order to achieve limit-sure selective termination on OC-MDPs, it is sufficient to play using a memoryless-deterministic strategy. We show that the *almost-sure* $\{1,2,3\}$*-parity* problem for OC-MDPs is at least as hard as the *limit-sure selective termination* problem for OC-MDPs.

The first result has been published in [5], where the author of this thesis played a significant role in the development of key ideas as well as proving formally results, such as:

- construction of the Timed Petri Nets model
- proving that the existential coverability problem for Timed Petri Nets is PSPACE-hard (lower bound)
- establishing relationship in terms of existential coverability problem between a general Timed Petri Net and a non-consuming Timed Petri Net (Lemma 2.7.1)
- construction of the region abstraction for markings of a Timed Petri Net
- proving lemmas regarding timed steps (Lemma 2.7.7, Lemma 2.7.8) as well as key idea in discrete step lemma Lemma 2.7.4
- construction of the acceleration procedure as well as proof of termination and key ideas and formal arguments about correctness of Algorithm 1

The second result has been published in [1], where the author of this thesis played a primary role in showing that the limit-sure control state reachability for single-sided VASS-MDPs where only the controller can change counter values (i.e., 1-VASS-MDPs) is decidable. The primary (and original) contribution of this result is based on the procedure of reducing the dimension of a 1-VASS-MDP (Algorithm 2), along with its termination and correctness. The third result has been published on its own, where the author of this project has provided the construction as well as the main result (Theorem 3.4.7), along with different lemmas that are directly linked to it.

1.2 Outline of the Thesis

The main goal of this thesis is to study decidability and complexity problems on certain classes of finitely representable systems whose underlying structure induce infinite state transition systems with stochastic or timed behaviour. In Chapter 1 we present a short introduction about these systems and types of problems that we study, along with a review with some well known results from this field.

In Chapter 2, we study the *Existential Coverability* problem on Timed Petri Nets (TPNs). These systems are a (timed) extension of classical Petri nets where every token has a real-valued clock (also known as age), whereas transition firing is constrained by the clock values that have integer bounds (using strict or non-strict inequalities). There exist several models, depending on what happens to the clock values of the newly

created tokens. We consider the general case, in which a newly created token can either inherit the age from an input token of the transition, or it can be reset to zero.

We will introduce the preliminaries required to understand the mathematical framework of Timed Petri Nets, along with the details regarding the problem that we are solving, as well as how does the model that we study relates to other models, such as Timed Networks, Timed Automata, or Priced Timed Petri Nets. A timed network consists of an arbitrary number of initially identical 1-clock Timed Automata, which interact via handshake communication. Hence, a Timed Petri Net is equivalent to a distributed (timed) network without a central controller, since initially there does not exist any tokens on other places that may be used to simulate one. Among the most well studied decision problems are *Reachability* and *Coverability*. The *Reachability* problem asks whether given a starting marking, it is possible to reach a certain marking. The *Coverability* problem asks whether given an initial marking, it is possible to reach a marking that enables a certain transition. It has been shown that the reachability problem is undecidable for Timed Petri Nets and all of its variants [83], whereas Coverability is decidable using a well-quasi-ordering approach as in [10], and complete for the complexity class $\mathcal{F}_{\omega^{\omega^{\omega}}}$. We show that the *Existential Coverability* problem for a Timed Petri Net is decidable and PSPACE-complete. Our main motivation for studying this problem lies in the fact that it corresponds to checking safety properties in distributed networks of arbitrarily many initially identical timed processes that communicate by handshake. A negative answer of the *Existential Coverability* problem certifies that a bad event of enabling a certain transition can never happen. Note that this is the *dual* problem of the *universal safety* problem, which asks whether from a starting marking it will always produce a good outcome. The full mathematical model will be provided in Chapter 2.

Chapter 3 is split between two related (but nevertheless different in terms of computation) probabilistic systems. We start by providing some preliminaries required to understand the rest of the chapter, in particular MDPs/MCs and perform an analysis on the types of problems on them, such as qualitative and quantitative ones. We then perform a review on probabilistic systems, such as Probabilistic Vector Addition with States (PVASS) along with several decision problems that have been recently studied. The first model that we study is a (probabilistic) extension of the VASS model, which we will call VASS-MDPs. There exists an extensive literature of algorithmic problems for VASS; however, one would like to study problems such as reachability on this probabilistic variant. For general VASS-MDPs, we show that even the simplest of the

probability-1 qualitative problems such as (almost)-sure reachability, is undecidable. Hence, we consider two monotone subclasses of VASS-MDPs, which are called *single sided* VASS-MDPs, in which either the controller, or the probabilistic entity can unilaterally change the counter values, but not both. We show that limit-sure control state reachability for single-sided VASS-MDPs where only the controller can change the counter values is decidable. We construct an algorithm and prove its correctness, where at each iteration the algorithm reduces the dimension of the VASS, while at the same time preserving the limit sure reachability properties.

We study an algorithmic problem called the *limit-sure selective termination* problem on One-Counter Markov Decision Processes (OC-MDPs), a probabilistic model that extends finite state MDPs with an unbounded counter. In a OC-MDP, the counter may be incremented, decremented, or left unchanged depending on the current control state or whether the counter is zero or not. Note that this model is equivalent to a controlled extension of discrete time Quasi-Birth-Death (QBDs) processes, a system that is studied in queueing theory. Also, OC-MDPs are mathematically equivalent to a probabilistic extension of One-Counter Automata (OCAs), and subsume a particular class of MDP models called solvency games [20], a model in which a controller (also known as gambler) would like in the long run behaviour to never become bankrupt. In a OC-MDP, the limit-sure selective termination problem asks whether starting from an initial control state with counter value 1 it is possible to reach a set of states (a.k.a. targets) with counter value zero via an infinite family of strategies with probability arbitrarily close to 1.

Our motivation for studying this problem comes from [28], where it has been left open. The limit-sure case is different from its corresponding almost-sure case, since more complicated behaviours in the structure of strategies may occur. In general, termination problems for OC-MDPs (i.e., reaching counter value zero) are not only interesting for the theoretical aspect, but they might be applied to real-world scenarios such as checking battery life and energy levels of hardware components, or financial models. However, the limit-sure case of *selective* termination does not have directly applications. Nevertheless, it is still a hard problem and this thesis establishes a connection with a subcase of the parity problem, called the $\{1,2,3\}$-parity problem. In order to define the $\{1,2,3\}$-parity problem for an OC-MDP, one all control states of the system are colored, i.e., they are labelled by a natural number. The almost-sure $\{1,2,3\}$-parity problem asks whether with probability 1 the maximal color which is visited infinitely often is 2.

One does not claim that specifically the $\{1,2,3\}$-parity can be directly applied to real-world scenarios. Note that $\{1,2,3\}$-parity is just a small subcase of general parity, but already very hard (unlike simpler subcases of parity in the Mostowski hierarchy; see [61]).

1.3 Introduction to (Timed) Petri Nets

In this section we are going to introduce a timed extended class of Petri Nets, also known as Timed Petri Nets, along with its decision problem that we are going to study, namely the *Existential Coverability* problem. We give a brief overview of different models that are present in the literature, such as classical Petri Nets, Timed Networks, Petri Nets with Time and Cost (a.k.a. *Price Timed Petri Nets*) and Timed Automata.

First we will introduce the concept of (classical) Petri Nets, a well studied model that gave many fruitful research results.

1.3.1 Petri Nets

Petri nets were firstly introduced by Carl Adam Petri in 1962 in his PhD thesis [75], having been applied to a wide variety of areas, such as distributed systems [82], systems biology [36], networking [81], etc.

A *Petri net* (PN) provides a mathematical model for the description and representation of distributed systems. It is represented as a directed (bipartite) graph - the set of nodes being partitioned into *places* and *transitions*, whereas the set of (directed) edges are called *arcs*, connecting places and transitions. The directed arcs represent which places are *preconditions* or *postconditions* for which transitions. A directed arc may connect a place with a transition or vice-versa, and it cannot connect two places or two transitions. A place from which an arc points to a transition is called an *input* place. Conversely, a place to which the arc points from a transition is called an *output* place. Each place may contain a (discrete) number of *tokens*. A *marking* is a distribution of tokens over the corresponding places of the Petri Net. A transition is said to be *enabled* if there exists sufficiently many tokens on the input places. When firing a transition, some tokens are *consumed* from the input places and new tokens may be *created* on the output places. See Example 1.3.1.

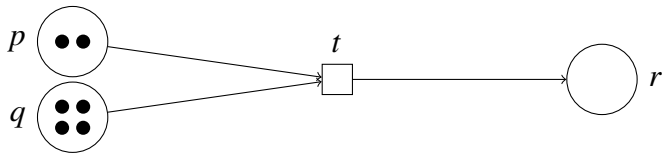
There exist multiple subclasses of Petri nets, such as *conflict-free*, *one-safe*, *cyclic*, *persistent*, etc. A complete review of their properties and their decision problems can

be found in [39]. A marking M is said to be *reachable* from an initial marking M_0 if there exists a sequence of transition steps which start at M_0 and end up in M. There exist multiple decision problems that have been studied in the literature, among which we recall:

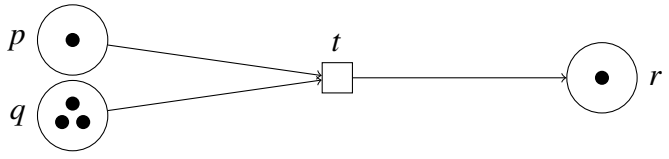
- The *Reachability* problem: Given a Petri Net $\mathcal{N}$ and 2 markings M_0 and M_f, one would like to know whether there exists a sequence of transition steps which start at M_0 and permit us to reach M_f.
- The *Coverability* problem: Given a Petri Net $\mathcal{N}$ and 2 markings M_0 and M, one would like to reach from M_0 a marking M' which *covers* M (M' is larger than M, with $M' \geq M$).
- The *Boundedness* problem: Given a Petri Net $\mathcal{N}$, one would like to know whether the set of reachable markings is finite. $\mathcal{N}$ is said to be k-bounded if places never hold more than k tokens. Moreover, $\mathcal{N}$ is called *safe* if places hold at most one token.
- The *Liveness* problem: Given a Petri Net $\mathcal{N}$, one would like to know whether for every reachable marking M and every transition t, there exists a sequence of transitions from M to M', such that M' enables t.

Another decision problems studied in the literature are the *deadlock-freedom* problem, the *home state* problem, the *promptness* and *strong promptness* problem, as seen in the complete literature review from [39].

Example 1.3.1. *The picture below shows a Petri net representation with three places p, q, r and one transition t.*



Now, by firing transition t, the Petri Net becomes



The transition t consumes one token from place p, and one token from place q, producing a new token on place r.

Now we will extend the Petri Net model by introducing time, and equip tokens with (real) clock values. This new model is called a Timed Petri Net.

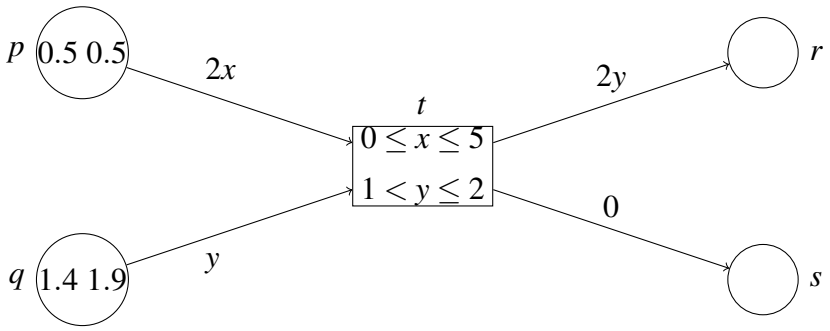
1.3.2 Timed Petri Nets

Our model of a Timed Petri net is a Petri net in which every token has a real-valued clock This information is also known as *token age*. There exists two types of transitions, such as *discrete* and *timed*. In firing a *discrete* transition, the tokens which are removed from the input places must have clock values in the interval of its corresponding (input) arc. Similarly, the tokens which are newly produced on the output places will have clock values either reset to zero or can inherit the clock values of some tokens from the input places. Note that this is a restriction that we will preserve for our decision problem. In general, this model can also be extended by allowing newly created tokens to have any clock value in a specified interval. We assume a *lazy* behaviour of the Timed Petri net, where firing of the transitions may be delayed, even if that will imply that certain transitions may become disabled due to the fact that the tokens on the corresponding input places are too old. Every *discrete* transition has a *transition guard*, which models an interval that clock values may take in order to fire the transition. In order to do this, we use transition variables that map to concrete clock values. See Example 1.3.2 for a concrete example.

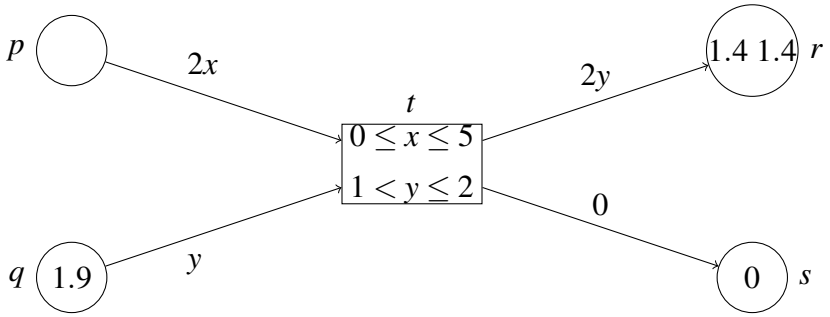
In a *timed* transition, all clock values of the tokens are incremented by the same real amount.

Example 1.3.2. *The picture below shows a representation of a Timed Petri Net with four places p,q,r,s and one transition t, with transition variables x and y. Consider variable x maps to* 0.5 *and variable y maps to* 1.4.

The way we interpret the discrete transition firing is the following. Transition t consumes two tokens (from place p) whose clock value is mapped to x (i.e. 0.5*) and one token (from place q) whose clock value is mapped to y (i.e.* 1.4*). The transition may fire (i.e is enabled) only if there are sufficiently many tokens on the input places, and they satisfy the transition constraint guards. Namely, since* $0 \leq 0.5 \leq 5$ *and* $1 < 1.4 \leq 2$*, then the transition will be enabled.*

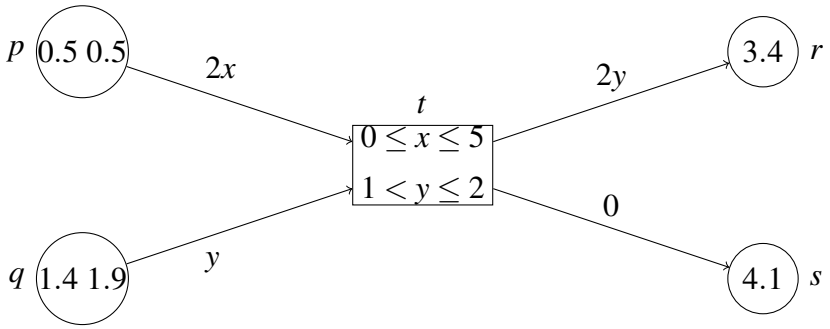


After firing t, we will obtain

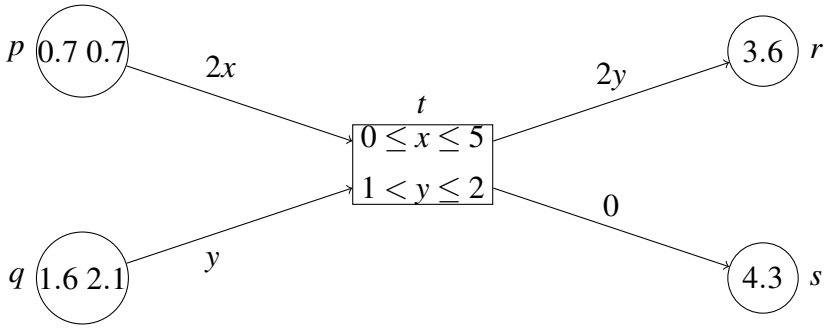


A *timed* transition is represented in Example 1.3.3.

Example 1.3.3. *The picture below shows a representation of a Timed Petri Net with four places* p,q,r,s *and one transition* t*, with transition variables* x *and* y*. Consider variable* x *maps to* 0.5 *and variable* y *maps to* 1.4*.*



Let us consider that all clocks will age by 0.2*. Hence, we have that*



As in the Petri net model, a marking represents a distribution of tokens (along with their clock values) in the corresponding places.

The *Reachability*, *Coverability*, *Boundedness*, *Liveness* problems are defined similarly as for classical Petri nets.

Now we introduce our decision problem that we are going to solve, namely the *Existential Coverability* problem.

1.3.3 Existential Coverability Problem

Given a Timed Petri Net $\mathcal{N}$, a place p of $\mathcal{N}$ and a transition t, the *existential coverability* problem asks whether there exists a number $m \in \mathbb{N}$ of tokens such that starting with m tokens of age 0 on place p (with all other places being empty), transition t is eventually enabled. I.e., whether there exists a finite sequence of transitions towards a marking that allows transition t to be fired. We recall that in the TPN framework a transition is enabled if there are sufficiently many tokens in the corresponding input places, along with satisfying the time constraints referring to their clock values.

We will show in Chapter 2 that this problem is decidable and PSPACE-complete.

1.3.4 Related Models

We recall time based systems which are related to the model of Timed Petri Nets, such as

- *Timed Networks* [6] - A *Timed network* (TN) represents a family of infinitely many systems, where each system is made of a *controller* and an arbitrary number of same timed processes. The controller is made by a finite state, whereas each process is a Timed Automaton. In each process, every clock value increases continuously at the same rate. Further details can be found in Section 2.4.2.

- *Timed Automata* [12] - A *Timed Automaton* is a finite automaton which is equipped with a finite set of real-valued clock variables, also known as *clocks*. Along a run of a timed automaton, clock values increase all at the same speed, and transitions may be enabled or disabled according to *clock guards*. Equivalently, they are also known as *clock constraints*. During a run, clocks can be inspected or reset to zero. Further details can be found in Section 2.4.1.

- *Price Timed Petri Nets* [4] - A Price Timed Petri Net is a Timed Petri Net equipped with a *cost* function among transitions.

- *Petri nets with Data* [63] - A Data Net is a generalisation of Petri nets where tokens have data from linearly-ordered infinite domains that also support whole-place operations, like resets and transfers.

1.4 Introduction to Probabilistic Systems

In this section, we will provide a basic introduction to probabilistic systems along with their corresponding related work and results that are relevant to our models, in particular VASS and OC-MDPs. We do not provide a mathematical framework here, since this will be presented in Chapter 3. However, we would like to present the intuition behind what we have achieved and how these results fit according to other models.

1.4.1 Markov Chains (MCs) and Markov Decision Processes (MDPs)

The main building blocks for our algorithmic analysis that we are going to study are the Markov Chains (MCs) and Markov Decision Processes (MDPs).

As in [73], MCs are processes used to model systems that evolve in time under uncertainty. In other words, it is a mathematical system in which it can experience transitions from one state to another based on stochastic rules. The core property of a Markov chain lies in the fact that it does not matter how the system arrived in the present state (i.e its *history*), and the possible future states are fixed. Hence, the probability of going to any particular state is dependent only on the current state and time elapsed. In the literature, this is known as the *Markov property*. Markov chains occur in a variety of fields that deal with uncertainty, such as game theory [72], economics and finance [66]. If the probability of any state transition is independent of time, the Markov chain is *time-homogeneous*. This process can be represented as a labelled directed graph, for which the sum of the labels of any vertex's outgoing transitions is 1.

Example 1.4.1. Given a Markov chain with 2 states p, q, with the probabilities listed on its transitions/edges as in Figure 1.1, one would like to determine what is the probability of a process starting at p will be at q after 2 moves. Clearly, in order to transition from p to q, the system may either in the first move loop back to p and go to q in the second move, or move to q in the first move and stay there, i.e loop back to q. Hence, from basic probability arithmetic, we have that the probability of a process starting at p will be at q after 2 moves is in fact $0.4 \times 0.6 + 0.6 \times 0.3 = 0.42$.

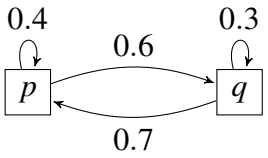


Figure 1.1: A time-homogeneous (finite) Markov Chain

In what follows, we will refer to systems on discrete time. For completeness, we will summarize some basic properties of Markov Chains as stated in [73].

- The period k of a state p is the greatest common denominator of all integers $n \geq 0$ for which the probability of starting in p and return in p after n states is strictly positive. In the case where $k = 1$, p is called *aperiodic*, whereas if $k > 1$, p is called *periodic*. A Markov chain is aperiodic if all of its states are aperiodic.

- A Markov chain is said to be *irreducible* if there exists a sequence of steps between any two states that has strictly positive probability.

- A state q is said to be *absorbing* if the probability of going from p to p in one step is 1.

- A state p is said to be *recurrent* if starting from p the probability of visiting p in the long run is 1. A recurrent state is called positive recurrent if the expected time to return to state p is finite. Conversely, if the expected time to return to state p is infinite, p is said to be *null recurrent*. Otherwise, if the probability of *never returning* to p is *stricty positive*, the p is called *transient*.

- We call a state p *ergodic* if it is aperiodic and positive recurrent. Hence, a Markov chain in which all of its states are ergodic is called ergodic.

A *Markov Decision Process (MDP)* is a directed graph made of a countable set of vertices (also known as *states*), equipped with a set of edges (also known as *transitions*). The set of states is partitioned into two disjoint sets - one that specifies which states belong to a *controller* (or non-deterministic player) and one which denotes the states where stochastic behaviour is exhibited (also known as *probabilistic* states). Moreover, the set of transitions refer to the type of states described, and greatly influence the way transitions are chosen. If a state is controlled, then any outgoing transition can be chosen non-deterministically. Conversely, if a state is probabilistic, there exists a rational probability distribution over the set of outgoing transitions. Intuitively, a

strategy is a recipe for the controller to choose what transitions to take at states that he controls. In this terminology, a *play* is a path (which may be finite or infinite) which is produced via states and transitions that occur according to a strategy and the probabilistic behaviour. An infinite play is called a *run*. MDPs are also called $1\frac{1}{2}$-player game, since the non-deterministic player would like to play against a stochastic environment (or nature) in order to achieve a certain objective.

1.4.2 Objectives on Probabilistic Systems

Given a MDP or MC M, an *objective* denotes a property on the set of plays of M, which can take several forms. Here, we mention some of the most encountered objectives that occur in the literature. Given a subset T of states of M, we call T to be a *target set*. We say that T is *reached* if there exists a play that visits *at least one* state in T.

- The *reachability* objective is made of the set of plays that end up in some state/set of states (a.k.a. *target set*) which is given a priori. It is one of the most frequently studied objective, which can occur in combination with other properties.

- The *repeated reachability* objective is made of the set of plays that visit some given target set *infinitely often*. This objective is also known as the *Büchi* objective.

- The *parity* objective is made of the set of infinite plays that satisfy the *parity* condition. On MDP/MC M, one would define a *color* (also known as *priority*) function, which labels every state of M with a natural number. A play satisfies the *parity* condition if and only if the minimal color (represented as a natural number) that is visited infinitely often among the states of M is *even*. Note that the *Büchi* objective (a.k.a. repeated reachability) is a special class of parity objective, where the states of M are labelled with numbers in $\{1,2\}$. Conversely, the co-*Büchi* objective is a special class of a parity objective, where the states of M are labelled with number in $\{0,1\}$.

- The *energy* objective and the *mean-payoff* objective are defined on a finite MDPs with a reward function where transitions are labelled with an integer. In an energy condition, the goal for the controller is to forever keep the value of the accumulated reward non-negative (never drop below zero). The mean-payoff condition requires that the limit-average of the accumulated reward to be within a threshold (usually strictly positive).

In a similar manner, one can define objectives on a non-stochastic transition system as well.

Given an MDP M, and a state c of M, we define the *value* at state c w.r.t. objective *Obj* by considering all probabilities of satisfying *Obj* using each possible strategy starting from state c and taking the supremum of them. A strategy σ is said to be optimal starting at a given state c if using the moves of σ the objective *Obj* is satisfied with probability equal to the value at state c w.r.t. *Obj*. Further details will be provided in Chapter 3.

1.4.3 Types of Objective Analysis on MDPs/MCs

W.l.o.g., let us consider M to be a countably infinite state MDP [3], and let us consider an arbitrary objective E. An analysis of an objective can be classified into two distinct classes, such as *qualitative* and *quantitative*. Qualitative properties can be split into several subclasses, such as *sure*, *almost-sure*, and *limit-sure*.

1. Qualitative:
   - the objective is satisfied *surely*: all runs induced by every strategy σ of M satisfy the objective.
   - the objective is satisfied *almost surely*: there exists a strategy σ on M such that the probability of achieving the objective using σ is 1.
   - the objective is satisfied *limit surely*: there exists an infinite family of strategies that satisfy the objective with probability arbitrarily close to 1.
2. Quantitative:
   - *Exact*: one would like to find a strategy σ, compute the probability p using σ of achieving a certain objective *Obj* and decide exact questions, such as whether $p \oplus \alpha$, where α is a constant and $\oplus \stackrel{\text{def}}{=} \{<, >\}$.
   - ε-*approximation*: one would like to ε-approximate the *value* of a given starting state v w.r.t. objective *Obj*, denoted as $Val(v)$. I.e., one would like to compute p' such that $p' \leq Val(v) \leq p' + \varepsilon$. Typically, each inequality is witnessed by some strategy (and we might sometimes only approximate the attainment of some of the witnessing strategies, as found in [40]-[45]).

[3] Note that a MC is an MDP whose set of *controlled* states is *empty*

1.4.4 Why Finite State Methods do not Work for Infinite-State Systems

It is important to note from the beginning that many techniques that apply to solving or optimizing certain objectives on finite state MDPs do not carry over to infinite state MDPs. In the case of a finite state MDP, there always exists a minimal value different from zero. However, for infinite state MDPs, this may not be case. For example, consider the reachability objective, and an infinite state MC as in Example 1.4.2, that represents the classical gambler's ruin problem (as stated in [73]).

Example 1.4.2. The gambler's ruin problem is a very well known problem in random walk theory. Informally, a gambler will start with an initial wealth of 1 dollar and then onwards, on each successive gamble it can either win 1 dollar or lose 1 dollar independent of the past, with fixed probability p and $q = 1 - p$, respectively. Once the gambler has gone bankrupt, i.e., has reached 0 dollars, he cannot do anything. This problem can be modelled as a Markov Chain M with infinite state space where the vertices are represented by numbers which denote gambler's wealth. The edges are labelled by the fixed probabilities p and q, representing a win or a loss, accordingly.

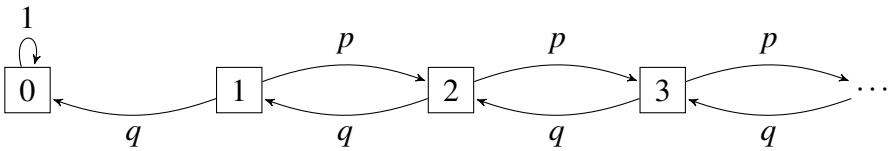


Figure 1.2: Gambler's ruin problem

In the biased gambler's ruin problem with unfair coin toss in gambler's (player's) favor, let the probability of winning 1 dollar is $p > \frac{1}{2}$. Since in this model every state of M is transient apart from the ruin state itself, it holds that the probability of ruin, i.e reach state 0, is > 0, for every vertex. Moreover, for every other state different from 0, the probability of ruin is < 1.

Also, another reason why standard techniques for solving/deciding problems on finite state MDPs are not suitable for solving problems on infinite state MDPs is the fact that optimal strategies may not exist, even for qualitative objectives such as reachability or parity. Hence, even if there exists a state whose value is 1, it may not be the case that there is a unique strategy that achieves value 1. However, there may exist an infinite family of strategies which are ε-optimal, for every $\varepsilon > 0$. In our work, we focus on

countably infinite MDPs that are *finitely branching*, where each state has a finite number of successor states.

The main textbook for studying general techniques which includes a very comprehensive mathematical framework on finite state MDPs is [76]. Recently, several classes of infinite state systems has been considered as well. For example, there exist studies of infinite state MDPs and stochastic games induced by finitely representable systems, such as one-counter systems (OC-MDPs [28] and OC-SSGs [26]).

1.4.5 VASS

Introduced in 1969 by Karp and Miller in [60], a VAS (Vector Addition System) provides a mathematical model for distributed systems. A Vector Addition System with States (VASS) is a generalized model of vector addition systems (VAS), which was introduced by John Hopcroft and Jean-Jacques Pansiot in 1979 [58].

Informally, a VASS is a directed graph made by a finite set of *control states* , which operate on a *n-dimensional vector of counters*. A counter can have a natural number value. The edges are labelled by n-dimensional integer vectors, also known as *counter operations* or *transitions*. The condition imposed on the counter values consists in the fact that they cannot drop below zero. Hence, some transitions may be *disabled*. Note that VASS are *non stochastic* models, since every available transition can be chosen deterministically. However, they provide a framework for the models that are probabilistic, such as VASS-MDPs.

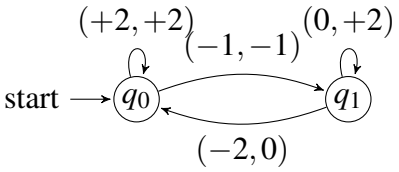


Figure 1.3: An example of a VASS with control states q_0 and q_1, with the counter operation listed on its edges.

In Figure 1.3, note that starting from control state q_0 and vector of counter values $(0,0)$, one can reach control state q_1 by first performing the loop transition towards q_0, and then move to q_1. At q_1 the vector of counter values is $(1,1)$.

There exists several decision problems that has been studied. For instance, in [77] it is shown that *control state reachability* for VASS is EXPSPACE-complete, reachability

in VASS is decidable [67], using a non-primitive recursive algorithm, whereas repeated control state reachability in VASS is shown to be EXPSPACE-complete ([53]). Note that VASS and Petri nets are mathematically equivalent [80].

1.4.6 PVASS

Probabilistic Vector Addition with States (PVASS) are stochastic extensions on VASS, where the control states possess probabilistic behaviour only. In [7], it has been shown that most *quantitative* objectives in PVASS are undecidable, or the solution not constructible in Presburger arithmetic. Moreover, the qualitative (almost sure) repeated reachability problem is shown to be decidable, provided that the set of target control state is upward-closed. The case where one can reach infinitely many times a set of target states with probability 0 has still been left open. Further details can be found in Chapter 3.

1.4.7 VASS-MDPs

VASS-MDPs are extensions of PVASS with non-deterministic choices that are made by a controller, as was firstly introduced in [1]. For general VASS-MDPs, we show that even the simplest of these problems, such as (almost)-sure reachability, is undecidable (Section 3.2.3). These systems subsume two monotone subclasses, such as 1-VASS-MDPs and P-VASS-MDPs, which are called *single sided*. In 1-VASS-MDPs, only the controller may change the counter values. In other words, on all control states owned by the random player the outgoing transitions leave the counter unchanged. In P-VASS-MDPs, this behaviour is swapped, namely that only the stochastic player may change the counter values.

For P-VASS-MDPs, all sure/almost sure/limit sure (repeated) reachability are undecidable. In the absence of deadlocks (i.e., paths that do not continue any further), the sure reachability/ repeated reachability become decidable. However, for 1-VASS-MDPs, the sure/almost sure/limit-sure reachability problem and sure/almost sure repeated reachability problem are *decidable* [1].

In particular, we show that the limit-sure control state reachability for 1-VASS-MDPs is decidable (Theorem 3.3.5).

1.4.8 One-Counter Markov Decision Processes (OCMDPs)

OC-MDPs are probabilistic variants of One-counter Automata, which in turn are extensions of finite state automata with an unbounded counter. An OC-MDP is made by a finite set of control states, that are partitioned into stochastic and controlled. On each transition, the counter can be decreased/increased by 1 unit or leaved unchanged.

The main motivation of studying this models comes from the work of [28], which study objectives such as termination and selective termination. Informally, a termination objective requires to reach counter value zero in any control state, whereas the selective termination is more stringent, since it requires to reach counter value zero while at the same time being in a set of control states which is given a priori (target states).

For a given OC-MDP, the *limit-sure selective termination problem* requires that starting from a given control state with counter value 1, one would like to reach counter value 0 in a given target set with probability arbitrarily close to 1.

In a $\{1,2,3\}$-*parity* problem, every state is colored with a natural number in $\{1,2,3\}$. The almost-sure $\{1,2,3\}$-*parity* problem asks whether there exists a strategy such that the highest color that is visited infinitely often is even.

We will show that almost sure $\{1,2,3\}$-parity problem for OC-MDPs is at least as hard as the limit-sure selective termination problem for OC-MDPs. Further details will be provided in Section 3.4.7.

1.4.9 Solvency Games

Solvency games (studied in [20]) model a risk-averse gambler (also known as *investor*). They are a subclass of OC-MDPs, since they have a single control state, but there may exist several actions that can modify the counter value (also known as *bankroll*). Further details can be found in Section 3.4.2.

1.4.10 Recursive Markov Chains (RMCs) and Recursive Markov Decision Processes (RMDPs)

Recursive Markov Chains (RMCs) denote a class of countably infinite MCs that are constructed by adding a natural recursion feature to finite state MCs. It has been shown in [45] that adding recursion to stochastic systems provides an abstract model to represent probabilistic procedural programs. Further details can be found in Section 3.4.3.

1.4.11 One-Counter Simple Stochastic Games (OC-SSGs)

OC-SSGs are a subclass of 2 player zero-sum stochastic games played on transition graphs of one-counter automata. Further details can be found in Chapter 3.

1.4.12 Limitation of Our Models

The infinite state (finitely branching) MDPs that we study in this thesis are induced by VASS-MDPs and OC-MDPs, respectively, which are finitely representable probabilistic systems. In this sense, our model is somewhat restricted to a particular class of systems, but nevertheless they have their own interesting problems, a lot of them being still open.

Chapter 2

Timed Petri Nets

In this chapter we define a mathematical formalism for representing Petri Nets where every token has a single real-valued clock, also known as *age*. This model is called a *Timed Petri Net*.

We introduce the model of Petri Nets, a model which has been researched since the '60s, along with the most frequently studied decision problems. We will present the mathematical formalism of Timed Petri Nets, perform a background survey about related (timed models), along with their decision results. We then present our main contribution, namely that the *Existential Coverability* problem for Timed Petri Nets is decidable and PSPACE-complete.

A *safety* property requires that nothing bad happens during the execution of a system. Usually, one can study safety properties by first characterizing states/configurations/-markings that we do not want to occur, considered to be *bad*. In the Petri net framework, safety properties can be modelled by checking whether from an initial marking one can fire a sequence of transitions to a (set of) bad markings.

2.1 Notation

We use $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ to denote the sets of nonnegative integers and reals, respectively. For $n \in \mathbb{N}$ we write $[n]$ for the set $\{0, \ldots, n\}$.

For a set A, we use A^* to denote the set of words, i.e. finite sequences, over A, and write ε for the empty word. If R is a regular expression over A then $\mathcal{L}(R) \subseteq A^*$ denotes its language.

A *multiset* over a set X is a function $M : X \to \mathbb{N}$. The set $X^{\oplus}$ of all (finitely supported) multisets over X is partially ordered pointwise (by $\leq$). The multiset union

of $M, M' \in X^{\oplus}$ is $(M \oplus M') \in X^{\oplus}$ with $(M \oplus M')(\alpha) \stackrel{\text{def}}{=} M(\alpha) + M'(\alpha)$ for all $\alpha \in X$. If $M \geq M'$ then the multiset difference $(M \ominus M')$ is the unique $M'' \in X^{\oplus}$ with $M = M' \oplus M''$. We will use an additive representation and write for example $(\alpha + 3\beta)$ for the multiset $(\alpha \mapsto 1, \beta \mapsto 3)$. For a multiset M and a number $m \in \mathbb{N}$ we let $m \cdot M$ denote the m-fold multiset sum of M. We further lift this to sets of numbers and multisets on the obvious fashion, so that in particular $\mathbb{N} \cdot S \stackrel{\text{def}}{=} \{n \cdot M \mid n \in \mathbb{N}, M \in S\}$.

Definition 2.1.1. A *pre-order* $(A, \preceq)$ is made of a set A and a reflexive and transitive relation $\preceq$ on A. If $\preceq$ is also symmetric, then $\preceq$ is an equivalence relation. A set $U \subseteq A$ is called *upward-closed* w.r.t. $\preceq$ if $c \in U$ and $c \preceq c'$ implies that $c' \in U$. For $a \in A$, we define $\uparrow a \stackrel{\text{def}}{=} \{b \mid a \preceq b\}$, i.e., $\uparrow a$ is the upward closure of a w.r.t. $\preceq$. For a set $B \subseteq A$, we define $\uparrow B \stackrel{\text{def}}{=} \bigcup_{a \in B} \uparrow a$. One can define downward-closed sets and downward closures in the similar way. A well-quasi-ordering (WQO) is any pre-order $\preceq$ over set A such that for any infinite sequence $a_0, a_1, \ldots$ in A, it is the case that there exist indexes $i < j$ such that $a_i \preceq a_j$.

2.2 Petri Nets

Definition 2.2.1. A *Petri Net* is a tuple $\mathcal{N} = \langle P, T, In, Out \rangle$, where P is a finite set of *places*, T is a finite set of transitions (with $P \cup T \neq \varnothing$ and $P \cap T = \varnothing$), In is a finite multiset over $P \times T$, Out is a finite multiset over $T \times P$.

A *marking* M of a Petri Net $\mathcal{N}$ is an assignment of tokens to the places of a Petri Net, i.e., it is a finite multiset over the set of places P. Intuitively, it represents the number of tokens in each place. A place p is *marked* by a marking M if $M(p) > 0$. We define the *null/empty* marking as the marking which maps every place to 0.

A transition t is said to be *enabled* if every input place p of t contains at least the number of tokens equal to the weight of the (directed) arc which connects p to t, namely that $M(p) \geq In(p,t)$, for all $p \in P$.

An enabled marking can be fired. The firing of an enabled transition t removes from every input place p a number of tokens equal to the weight of the (directed) arc which connects p to t and produce in every output place p the number of tokens which is equal to the weight of the (directed) arc that connects t to p.

Given two markings M and M' of Petri Net $\mathcal{N} = \langle P, T, In, Out \rangle$, we have that $M \longrightarrow M'$ if there exists a transition $t \in T$ such that

- $M(p) \geq In(p,t)$, for every $p \in P$

- $M'(p) = M(p) - In(p,t) + Out(t,p)$, for every $p \in P$

We define $\longrightarrow_*$ to be the reflexive and transitive closure of $\longrightarrow$.

A transition system induced by the Petri Net $\mathcal{N}$ is defined by the set of markings of $\mathcal{N}$ along with the transition relation on those.

We will illustrate some of the most frequently studied decidability questions for Petri Nets.

2.2.1 Decidability questions for Petri Nets

1. *Reachability* problem: Given a Petri Net $\mathcal{N}$, an initial marking M_0, and a marking M_f, one would like to decide whether $M_0 \longrightarrow_* M_f$. Intuitively, given an initial marking M_0 and a final Petri Net marking M_f we would like to know whether there exists a sequence of transition steps which start at M_0 and permit us to reach M_f. It has been shown to be decidable in [67].

2. *Coverability* problem: Given a Petri Net $\mathcal{N}$, and markings M_0, M, one would like to check whether there exists a sequence of transitions such that $M_0 \stackrel{*}{\longrightarrow} M'$, with $M \leq M'$. In other words, one would like to decide whether starting from initial marking M_0 one can reach via a sequence of Petri Net transitions a marking M' with $M' \geq M$. In this way, we say that marking M is *covered* by M'. Hence, due to its structure, the Coverability problem is equivalent to the problem of Reachability of an upward closed set of Petri Net markings. It has been shown to be *decidable* in [77].

3. *Boundedness* problem: Given a Petri Net $\mathcal{N}$, one would like to check whether its set of reachable markings is *finite*. This problem is *decidable*, using what is called the *coverability tree* method [60], by identifying places and tokens (i.e., token generators) from which the size of the Petri Net can become unbounded. The construction of the coverability tree is however inefficient, by requiring non-primitive recursive space. The work by Rackoff [77] gives an exponential space algorithm.

4. *Liveness* problem: Given a Petri Net $\mathcal{N} = \langle P, T, F \rangle$, for every reachable marking M and for every transition t, there exists a sequence of transitions such that $M \stackrel{*}{\longrightarrow} M'$, where M' enables t. This problem has been shown to be decidable [54].

A complete description of Petri Nets along with their subclasses and decision processes can be found in [39].

2.3 Timed Petri Nets

Timed Petri nets are place/transition nets where each token carries a real value, sometimes called its *clock value* or *age*. Transition firing depends on there being sufficiently many tokens whose value is in a specified interval. Timed-arc Petri nets (TPN) [10, 85, 8, 24, 59] are an extension of Petri nets where each token carries one real-valued clock and transitions are guarded by inequality constraints where the clock values are compared to integer bounds (via strict or non-strict inequalities). The known models differ slightly in what clock values newly created tokens can have, i.e., whether newly created tokens can inherit the clock value of some input token of the transition, or whether newly created tokens always have clock value zero. We consider the former, more general, case.

All tokens produced by a transition either have age 0, or inherit the age of an input-token of the transition. To model time passing, all token ages can advance simultaneously by the same (real-valued) amount.

There exist several mathematical formalisms, such as [78], [69], [10] for representing TPNs whose tokens have clock values and transition constraints.

Timed Petri Nets (TPNs) can be classified into two classes, according to the domain of the clock values of tokens. If the clock values are defined on a (non-negative) countable domain, then the net is called *discrete* Timed Petri Net. In other words, time is being incremented in discrete steps. On the other hand, if the clock values are defined on (non-negative) real numbers, then the net is called *dense* Timed Petri Net, where time is interpreted as continuous.

In our analysis, we consider the latter case, namely the model of dense Timed Petri Nets, since every token has a real value age. Moreover, the model of Timed Petri Nets we employ, which is similar to [10] , uses the fact that the system is unbounded, where the number of tokens that is situated on places of the net can grow indefinetely. Moreover, our transition firing rules employ a *lazy* semantics, namely that once a transition is enabled, it is not the case that it must fire. Hence, it might happen that as time passes (and tokens will age) that transition becomes disabled (due to transition constraint guards). Conversely, another semantics (which we will not use in our model)

is the *eager* firing transition semantics, in which transitions, once they are enabled, they have to be fired.

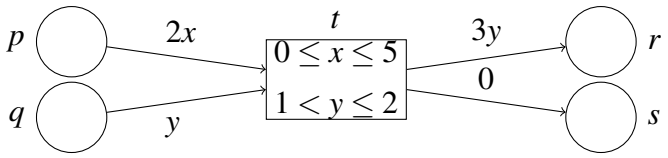
The TPN model corresponds to a controller-less Timed Network [6] of initially identical 1-clock Timed Automata which interact via handshake communication. Note that the techniques and properties for Timed Automata do not directly apply for TPN model, since on timed automata, the number of clocks is *finite*, whereas in our model there can be an unbounded number of clock values.

Definition 2.3.1 (TPN)**.** A *timed Petri net* (TPN) $\mathcal{N} = (P, T, Var, G, Pre, Post)$ consists of finite sets of *places* P, *transitions* T and *variables* Var, as well as functions $G, Pre, Post$ defining transition *guards*, *pre–* and *postconditions*, as follows.

For every transition $t \in T$, the guard $G(t)$ maps variables to (open, half-open or closed) intervals with endpoints in $\mathbb{N} \cup \{\infty\}$, restricting which values variables may take. The precondition $Pre(t)$ is a finite multiset over $(P \times Var)$. Let $Var(t) \subseteq Var$ be the subset of variables appearing positively in $Pre(t)$. The postcondition $Post(t)$ is then a finite multiset over $(P \times (\{0\} \cup Var(t)))$, specifying the locations and clock values of produced tokens. Here, the symbolic clock value is either 0 (demanding a reset to age 0), or a variable that appeared already in the precondition.

A *marking* for a TPN $\mathcal{N}$ is a finite multiset over $P \times \mathbb{R}_{\geq 0}$.

Example 2.3.1. *The picture below shows a place/transition representation of an TPN with four places and one transition.* $Var(t) = \{x, y\}$, $Pre(t) = 2(p, x) + (q, y)$, $G(t)(x) = [0, 5]$, $G(t)(y) =]1, 2]$ *and* $Post(t) = 3(r, y) + (s, 0)$.



The transition t consumes two tokens from place p, both of which have the same clock value x (where $0 \leq x \leq 5$*) and one token from place q with clock value y (where* $1 < y \leq 2$*). It produces three tokens on place r who all have the same clock value y (where y comes from the clock value of the token read from q), and another token with value* 0 *on place s.*

There are two different binary step relations on markings: *discrete* steps $\longrightarrow_t$ which fire a transition t as specified by the relations G, Pre, and $Post$, and *time passing* steps $\longrightarrow_d$ for durations $d \in \mathbb{R}_{\geq 0}$, which simply increment all clocks by d. We will define the discrete steps first, as in Definition 2.3.2.

Definition 2.3.2 (Discrete Steps)**.** For a transition $t \in T$ and a variable evaluation $\pi : Var \to \mathbb{R}_{\geq 0}$, we say that π *satisfies* $G(t)$ if $\pi(x) \in G(t)(x)$ holds for all $x \in Var$. By lifting π to multisets over $(P \times Var)$ (respectively, to multisets over $(P \times (\{0\} \cup Var))$ with $\pi(0) = 0$) in the canonical way, such an evaluation translates preconditions $Pre(t)$ and $Post(t)$ into markings $\pi(Pre(t))$ and $\pi(Post(t))$, where for all $p \in P$ and $c \in \mathbb{R}_{\geq 0}$,

$$\pi(Pre(t))(p,c) \stackrel{\text{def}}{=} \sum_{\pi(v)=c} Pre(t)(p,v) \qquad \text{and} \qquad \pi(Post(t))(p,c) \stackrel{\text{def}}{=} \sum_{\pi(v)=c} Post(t)(p,v).$$

Intuitively, the discrete firing transition relation is defined using a variable evaluation which ultimately maps preconditions and postconditions (represented as finite multisets over $(P \times Var)$ and $(P \times (\{0\} \cup Var))$, respectively) to finite multisets over $P \times \mathbb{R}_{\geq 0}$. An intuitive example can be found in Example 2.3.1.

Definition 2.3.3 (Enabled transition)**.** A transition $t \in T$ is called *enabled* in marking M, if there exists an evaluation π that satisfies $G(t)$ and such that $\pi(Pre(t)) \leq M$. In this case, there is a discrete step $M \longrightarrow_t M'$ from marking M to M', defined as $M' = M \ominus \pi(Pre(t)) \oplus \pi(Post(t))$.

In order to illustrate the firing of discrete transition t from Example 2.3.1, let us consider $\pi(x) = 0.5$ and $\pi(y) = 0.6$. In other words, the variables x and y will be mapped to (concrete) clock values 0.5 and 0.6, respectively. Hence, we have that

- $\pi(Pre(t))(p, 0.5) = Pre(t)(p,x) = 2$
- $\pi(Pre(t))(q, 0.6) = Pre(t)(q,y) = 1$
- $\pi(Post(t))(p, 0.5) = \pi(Post(t))(q, 0.6) = 0$
- $\pi(Post(t))(r, 0) = Post(t)(r,y) = 3$
- $\pi(Post(t))(s, 0) = 0$

Now we will define the *timed* transition relation for TPNs.

Definition 2.3.4 (Time Steps)**.** Let M be a marking and $d \in \mathbb{R}_{\geq 0}$. There is a time step $M \longrightarrow_d M'$ to the marking M' with $M'(p,c) \stackrel{\text{def}}{=} M(p, c-d)$ for $c \geq d$, and $M'(p,c) \stackrel{\text{def}}{=} 0$, otherwise. We also refer to M' as $(M + d)$.

We write $\longrightarrow_T$ for the union of all timed steps, $\longrightarrow_{Disc}$ for the union of all discrete steps and simply $\longrightarrow$ for $\longrightarrow_{Disc} \cup \longrightarrow_{Time}$. The transitive and reflexive closure of $\longrightarrow$ is $\stackrel{*}{\longrightarrow}$. $Cover\,(M)$ denotes the set of markings M' for which there is an $M'' \geq M'$ with $M \stackrel{*}{\longrightarrow} M''$, i.e.,

$$Cover\,(M) \stackrel{\text{def}}{=} \{M' \mid \exists M'' \geq M'.M \stackrel{*}{\longrightarrow} M''\}$$

2.4 Related Work

2.4.1 Timed Automata

A Timed Automaton (TA) is a finite state automaton, namely a graph with finite set of nodes and edges that is equipped with a finite set of real-valued clocks. Along the edges (a.k.a transitions) of the automaton, clock values can be compared to integer values, forming clock constraints, which are called transition guards. These constraints can enable or disable transitions, hence modifying its behaviour. A TA can perform two types of transitions, such as *discrete* and *timed*. In a timed transition, all clock values are incremented by the same real valued amount. A configuration of a TA captures information about the control states and clock values. The *Control State Reachability problem* for Timed Automata is decidable and PSPACE-complete [12]. Other decision problems on Timed Automata found in [12] beyond the scope of this project include the *Emptiness* problem (decidable and PSPACE-complete), the *Language equivalence and language inclusion* problems (undecidable), the *Universality* problem (undecidable). A complete survey on Timed Automata can be found in [13].

We recall the technique of *finite partitioning*, a method developed in [12] for Timed Automata. Essentially, from the infinite state transition system induced by a Timed automata, one can derive an abstract finite state system using an equivalence relation on the set of states, where each new state in the abstract system is the representative of one equivalence class. Transitions between two states in this newly constructed system are defined in terms of transitions between two corresponding equivalence classes. The region abstraction technique can be interpreted as an instance of the finite partitioning method. In a region, one can ignore the exact value of the clocks, and store information about their integral parts only and the ordering of their fractional parts. This allows a more compact representation of infinite sets of configurations. Hence, numerous algorithms for solving questions on Timed Automata use regions as symbolic representations rather than using concrete configurations, since they are easier to design. Other variants for this method include *zones*, where each zone refers to several regions, providing a more efficient representation.

2.4.2 Timed Networks

A *Timed network* (TN) represents a family of infinitely many systems, where each system is made of a finite state *controller* along with finitely many timed processes.

Each timed process represents a Timed Automaton, hence it uses a finite number of local real clock values. The values of every clock increase continuously at the same rate. A configuration of a Timed Network with k clock stores information about the controller, finitely many timed processes, and about clock values. The configuration of the network can be changed according to a finite number of rules. Every rule represents a set of transitions where the controller and a fixed number of processes synchronize and change their states at the same time. A rule can have conditions on the local state of the controller as well as local states and the clock values of the timed processes. Once the conditions for a rule are satisfied, a transition can be performed. Then, the controller and every participating process will change its state. When performing a transition, a timed process may reset some of its clocks to 0.

Note that the techniques used for solving decidability questions for Timed Automata may not be applied directly for solving corresponding problems for Timed Network. The reason why this is the case lies in the fact that a Timed Network operates on an unbounded number of clocks, whereas for Timed Automata it is not the case. The behaviour for these models is therefore different.

In the case where each timed process has a single clock, the *Control State Reachability* problem for Timed Networks has been shown to be decidable [3]. The importance of this result relies on the fact that numerous classes of safety properties can be reduced to this problem. For multi-clock Timed Networks, it has been shown that even for the case where each timed process has only 2 clocks, the *Control State Reachability* is undecidable [6] (using a reduction from 2-Counter Minsky machines).

2.4.3 Timed Petri Nets

There exists an extensive literature on Timed Petri Nets, for both dense and discrete Timed Petri Nets. For the scope of our project, we will focus on the case of *dense* Timed Petri Nets. Several decision problems regarding (dense) Timed Petri Nets have been studied, such as:

1. The *Reachability* problem: As in the standard Reachability problem for classical Petri Nets, we have that for a given TPN $\mathcal{N}$, an initial marking M_0, and a marking M, one would like to check whether M is reachable via a sequence of (both timed and discrete) transition steps. It has been shown to be *undecidable* [37].

2. The *Coverability* problem: Given a TPN, and an initial marking M_0 and a marking M, the problem asks whether there exists a sequence of (discrete and timed)

transitions via a marking which covers M. It has been shown in [10] to be *decidable* using a backward reachability algorithm analysis, by using a symbolic representation (such as *existential zones*) for defining upward closed sets of markings. Moreover, the Coverability problem for TPNs is complete for the hyper-Ackermannian complexity class $F_{\omega^{\omega}}$ [55]. With respect to Coverability, TPNs are equivalent [21] to (linearly ordered) data nets [63].

3. The *Zenoness* problem: Given a TPN $\mathcal{N}$, and a marking M_0, one would like to check whether there exists an infinite sequence of transitions from M_0 with a finite duration only. It has been shown to be decidable [7] by using techniques from a subclass of *transfer Petri nets* , also known as *simultaneous disjoint transfer nets* [47].

4. The *Universal Zenoness* problem: Given a TPN $\mathcal{N}$ and a marking M, one would like to decide whether *every* infinite sequence of transitions takes only a finite amount of time. It has been shown [7] to be undecidable, by using a reduction from lossy counter machines.

5. *Token liveness* - Given a TPN $\mathcal{N}$ and marking M, one would like to check whether the token is not *dead*. A token is called *dead* if it can not be used in future computations. In [7] this problem has been shown to be decidable, by reducing it to the *Coverability* problem for TPNs.

6. *Boundedness* - Given a TPN $\mathcal{N}$ and marking M_0, one would like to check whether the size of reachable markings is finite. There exists two different decidability questions in terms of whether tokens which are not alive (i.e. dead) are taken into account. If dead tokens are taken as part of the size of the marking, then the problem is called *syntactic boundedness*. Conversely, if dead token are not taken as part of the size of the marking, the problem is called *semantic boundedness*. The *syntactic boundedness* problem has been shown in [7] to be decidable, using a modified Karp Miller algorithm where each node denotes a region (as opposed to a single marking as in the classical version). However, the *semantic boundedness* problem has been shown in [7] to be undecidable.

The *region abstraction* technique for TPNs is very similar to the region construction method for timed automata [12], where clocks are abstracted into finite regions, according to fractional clock values. Hence, one would store information about the clock

value of tokens in terms of fractional value and integral parts, separately. It extracts information about tokens whose fractional part is zero, tokens that have non-zero fractional part, and tokens which have the largest fractional value. Other variants include *existential zones* [10].

2.4.4 Priced Timed Petri Nets

A Priced-Timed Petri Net is a timed extension of Petri Nets in which every token has a real valued clock, and transition arcs are labelled with time intervals. Furthermore, tokens that are present on places have storage costs per unit of time, and transitions have firing costs. This model subsumes the TPN model as in [10], or Priced Timed Automata [19]. We recall that

- The Reachability problem with Minimal Costs is shown to be decidable (Theorem 13, [4])
- The Coverability problem with Minimal Costs is shown to be decidable, but it becomes undecidable if negative costs are allowed [4].

2.4.5 Timed-Arc Petri Nets

Timed-arc Petri nets are a (timed) extension of standard Petri nets, where time is continuous. Tokens in a timed-arc Petri Net will carry a clock value (age). This system has *arcs* between places and transitions which are labelled by intervals that restrict transition firing w.r.t. the age of tokens. A timed-arc Petri Nets can be *simple*, by having ordinary arcs, or can have other features, such as transport arcs or age-invariants [85]. Among the most important decidability questions for timed-arc Petri Nets, we recall the fact that

- The *Reachability* problem is *undecidable* [85] for even simple Timed-arc Petri Nets
- The *Coverability* and *Boundedness* problem are *decidable* for simple and Petri Nets with transport arcs. For other timed-arc Petri Net models, these problems are undecidable [37].

2.4.6 Petri Nets with Data

Data Nets offer a more generalized framework of Petri Nets where tokens have data from linearly-ordered infinite domains and in which whole-place operations are possible. Many classes of infinite-state models are subsumed by this framework, such as multiset rewriting systems, or polymorphic systems with arrays. In [63] it is shown that the coverability and the termination (i.e., whether all computations are finite) problems for arbitrary Data Nets are decidable. Moreover, it is shown that the boundedness problem is decidable for Data Nets where whole-place operations are restricted to transfers. Hence, related to our TPN, one can encode information about the real-valued clock ages, by allowing tokens to carry data from infinite domains.

2.5 Existential Coverability Problem for TPNs

We are interested in the *Existential Coverability problem* for TPNs ($\exists$COVER for short), as follows. The existential Coverability asks, for a given place p and transition t, whether there exists a number m such that the marking $M(m) \stackrel{\text{def}}{=} m \cdot \{(p,0)\}$ ultimately enables t. Here, $M(m)$ contains exactly m tokens on place p with all clocks set to zero and *no other tokens*. This problem corresponds to checking safety properties in timed distributed networks of arbitrarily many (namely m) identical timed processes that communicate by handshake. A negative answer certifies that the 'bad event' of transition t can never happen regardless of the number m of processes, i.e., the network is safe for any size. Thus by checking existential coverability, one solves the dual problem of *Universal Safety*.

The corresponding problem for Timed Networks studied in [6] does not allow the dynamic creation of new timed processes (unlike the TPN model which can increase the number of timed tokens), but considers multiple clocks per process (unlike our TPN with one clock per token).

The TPN model above corresponds to a distributed network without a central controller, since initially there are no tokens on other places that could be used to simulate one. Adding a central controller would make *Existential Coverability* polynomially inter-reducible with normal *Coverability* and thus complete for F_{ω^ω} [55] (and even undecidable for > 1 clocks per token [6]).

We recall that in the TPN framework a transition t is *enabled* in a marking M if there are sufficiently many tokens in the corresponding input places, along with satisfying

the time constraints. Formally, these two conditions are stated in Definition 2.3.3. Now we formally define our decision problem that we are going to solve, i.e., the *Existential Coverability problem* (denoted as ∃COVER for short) for TPNs.

Existential Coverability problem for Timed Petri Networks

Input: A TPN $\mathcal{N}$, an initial place p and a transition t.

Question: Does there exist $M \in Cover\,(\mathbb{N} \cdot \{(p,0)\})$ such that t is enabled in M?

Our main result lies in the fact that we show that ∃COVER is decidable and PSPACE-complete. Both lower and upper bound will be shown (w.l.o.g., see Lemma 2.7.1) for the syntactic subclass of *non-consuming* TPN, defined as follows.

Definition 2.5.1. A *timed Petri net* $(P,T,Var,G,Pre,Post)$ is *non-consuming* if for all $t \in T$, $p \in P$ and $x \in Var$ it holds that both 1) $Pre(t)(p,x) \leq 1$, and 2) $Pre(t) \leq Post(t)$.

In a non-consuming TPN, token multiplicities are irrelevant for discrete transitions. Intuitively, having one token (p,c) is equivalent to having an inexhaustible supply of such tokens.

The first condition is merely syntactic convenience. It asks that each transition takes at most one token from each place. The second condition in Definition 2.5.1 implies that for each discrete step $M \longrightarrow_t M'$ we have $M' \geq M$. Therefore, once a token (p,c) is present on a place p, it will stay there unchanged (unless time passes), and it will enable transitions with (p,c) in their precondition.

Wherever possible, we will from now on therefore allow ourselves to use the set notation for markings, that is simply treat markings $M \in (P \times \mathbb{R}_{\geq 0})^{\oplus}$ as sets $M \subseteq (P \times \mathbb{R}_{\geq 0})$.

2.6 Lower Bound

PSPACE-hardness of ∃COVER does not follow directly from the PSPACE-completeness of the reachability problem in timed automata of [12]. The reason why this happens is the fact that in our model we may allow an unbounded number of clocks, whereas timed automata operate on a finite amount of clocks only. Hence, the non-consuming property of our TPN makes it impossible to fully implement the control-state of a timed automaton. Instead our proof uses multiple timed tokens and a reduction from the iterated monotone Boolean circuit problem [49].

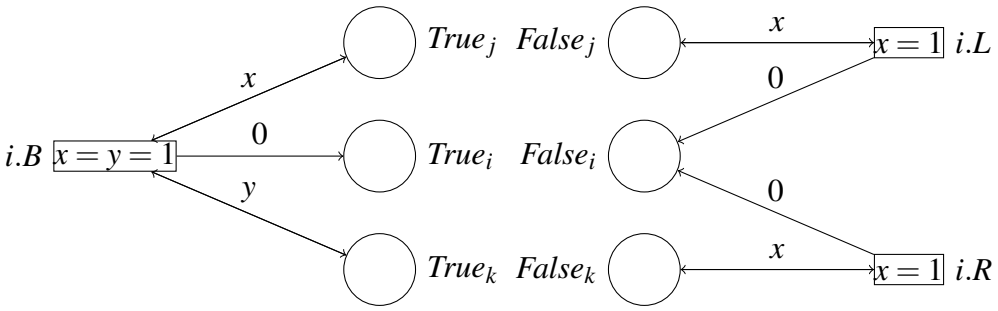


Figure 2.1: The transitions $i.B, i.R$ and $i.L$ that simulate the update of bit i according to constraint $i' = j \wedge k$. All transitions demand that incoming tokens are of age exactly 1 and only tokens of age 0 are produced.

Definition 2.6.1. A depth-1 monotone Boolean circuit is a function $F : \{0,1\}^n \to \{0,1\}^n$ represented by n constraints: For every $0 \leq i < n$ there is a constraint of the form $i' = j \otimes k$, where $0 \leq j,k < n$ and $\otimes \in \{\wedge, \vee\}$, which expresses how the next value of bit i depends on the current values of bits j and k. For every bitvector $v \in \{0,1\}^n$, the function F then satisfies $F(v)[i] \stackrel{\text{def}}{=} v[j] \otimes v[k]$.

Theorem 2.6.1. *[49] Given a vector $v \in \{0,1\}^n$, a depth-1 monotone Boolean circuit F, the problem of checking whether there exists a number $m \in \mathbb{N}$ such that $F^m(v)[0] = 1$ is* PSPACE*-complete.*

Towards a lower bound for ∃COVER (Theorem 2.6.3) we construct a non-consuming TPN as follows, for a given circuit. The main idea is to simulate circuit constraints by transitions that reset tokens of age 1 (encoding v) to fresh ones of age 0 (encoding $F(v)$), and let time pass by one unit to enter the next round.

For every bit $0 \leq i < n$, the net contains two places $True_i$ and $False_i$. A marking $M_v \leq P \times \mathbb{R}_{\geq 0}$ is an *encoding* of a vector $v \in \{0,1\}^n$ if for every $0 \leq i < n$ the following hold.

1. $(True_i, 0) \in M_v \iff v[i] = 1$.
2. $(False_i, 0) \in M_v \iff v[i] = 0$.
3. If $(p,c) \in M_v$ then $c = 0$ or $c \geq 1$.

Note that in particular one cannot have both $(True_i, 0)$ and $(False_i, 0)$ in M_v. For

every constraint $i' = j \wedge k$ we introduce three transitions, $i.L, i.R$, and $i.B$, where

$$Pre(i.B) \stackrel{\text{def}}{=} (True_j, x) + (True_k, y) \qquad Post(i.B) \stackrel{\text{def}}{=} Pre(i.B) + (True_i, 0)$$
$$Pre(i.L) \stackrel{\text{def}}{=} (False_j, x) \qquad Post(i.L) \stackrel{\text{def}}{=} Pre(i.L) + (False_i, 0)$$
$$Pre(i.R) \stackrel{\text{def}}{=} (False_k, x) \qquad Post(i.R) \stackrel{\text{def}}{=} Pre(i.R) + (False_i, 0)$$

and the guard for all transitions is $G(x) = G(y) = 1$. See Figure 2.1 for an illustration. For disjunctions $i' = j \vee k$ the transitions are defined analogously, with *True* and *False* inverted. The correctness proof of our construction rests on the following simple observation.

Lemma 2.6.2. If $F(v) = v'$ then for every encoding M_v of v, there exists an encoding $M_{v'}$ of v' such that $M_v \longrightarrow_1 \stackrel{*}{\longrightarrow}_{Disc} M_{v'}$. Conversely, if $M_v \longrightarrow_1 \stackrel{*}{\longrightarrow}_{Disc} M_{v'}$ for encodings M_v and $M_{v'}$ of v and v' respectively, then $F(v) = v'$.

Proof. For the first part, we construct a sequence $M_0 \longrightarrow_{Disc} M_1 \longrightarrow_{Disc} \ldots \rightarrow_D M_{n-1}$ where $M_0 \stackrel{\text{def}}{=} (M_v + 1)$ and every step $M_{i-1} \longrightarrow_{Disc} M_i$ adds tokens simulating the ith constraint of F. Since the TPN is non-consuming, we will have that $M_i \geq (M_v + 1)$, for all $i < n$. Consider now constraint i', and assume w.l.o.g. that $i' = j \wedge k$ (the other case is analogous). There are two cases depending on $v'[i]$.

1. Case $v'[i] = 1$. By our assumption that $F(v) = v'$ we know that $v[j] = 1$ and $v[k] = 1$. So $(True_j, 1) \in (M_v + 1) \leq M_{i-1}$ and $(True_k, 1) \in (M_v + 1) \leq M_{i-1}$. By construction of the net, there is a transition $i.B$ with $Pre(i.B) = (True_j, 1) + (True_k, 1)$ and $Post(i.B) = Pre(i.B) + (True_i, 0)$. This justifies step $M_{i-1} \longrightarrow_{i.B} M_i$ and therefore that $(True_i, 0) \in M_i \leq M_{n-1}$. Also notice that no marking reachable from M_0 using only discrete steps can contain the token $(False_i, 0)$. This is because these can only be produced by transitions requiring either $(False_j, 1)$ or $(False_k, 1)$, which are not contained in M_0 by assumption that M_v encodes v. Therefore $(False_i, 0) \notin M_{n-1}$.

2. Case $v'[i] = 0$. W.l.o.g., $v[j] = 0$. Therefore, $(False_j, 1) \in (M_v + 1) \leq M_{i-1}$. By construction of the net, there exists transition $i.L$ with $Pre(i.L) = (False_j, 1)$ and $Post(i.L) = Pre(i.L) + (False_i, 0)$. This justifies the step $M_{i-1} \longrightarrow_{i.L} M_i$, with $(False_i, 0) \in M_i \leq M_{n-1}$. Notice again that no marking reachable from M_0 using only discrete steps can contain the token $(True_i, 0)$. This is because these can only be produced by transitions $i.B$, requiring both $(True_j, 1), (True_k, 1) \in M_0$, contradicting our assumptions. Hence, $(True_i, 0) \notin M_{n-1}$.

We conclude that the constructed marking M_{n-1} is an encoding of v'.

For the other part of the claim, assume that there exist markings M_v and $M_{v'}$ which are encodings of vectors v and v', respectively, with $M_v \longrightarrow_1 \xrightarrow{*}_{Disc} M_{v'}$. We will show that $F(v) = v'$. Recall that $F(v)[i] \stackrel{\text{def}}{=} v[j] \otimes v[k]$, where $0 \leq j,k < n$ and $\otimes \in \{\wedge, \vee\}$. We will show for each $i < n$ that $v'[i] = v[j] \otimes v[k]$. Again, consider the constraint i', and assume w.l.o.g. that $i' = j \wedge k$ (the other case is analogous). There are two cases.

1. Case $v'[i] = 1$. By definition of a marking encoding, we have that $(True_i, 0) \in M_v$. By construction, there is a transition $i.B$ with $Pre(i.B) = (True_j, 1) + (True_k, 1)$ and $Post(i.B) = Pre(i.B) + (True_i, 0)$. By assumption, it holds that $(M_v + 1) \xrightarrow{*}_{Disc} M'_v$, where $M_v \longrightarrow_1 (M_v + 1)$. Note that $(True_j, 1) \in (M_v + 1)$ and $(True_k, 1) \in (M_v + 1)$. Hence, we have that $v[j] = 1$ and $v[k] = 1$, and therefore that $F(v)[i] = v'[i] = v[j] \wedge v[k]$.

2. Case $v'[i] = 0$. Then $(False_i, 0) \in M_v$ and, since this token can only be produced by transitions $i.L$ or $i.R$, either $(False_j, 1) \in (M_v + 1)$ or $(False_k, 1) \in (M_v + 1)$.

   Therefore $(False_j, 0) \in (M_v)$ or $(False_k, 0) \in (M_v)$ and because M_v is an encoding of v, this means that either $v[j] = 0$ or $v[k] = 0$. Therefore, $F(v')[i] = v[j] \wedge v[k] = 0$. □

Theorem 2.6.3. *$\exists$COVER is* PSPACE*-hard for non-consuming TPN.*

Proof. For a given monotone Boolean circuit, define a non-consuming TPN as above. By induction on $m \in \mathbb{N}$ using Lemma 2.6.2, we derive that there exists $m \in \mathbb{N}$ with $F^m(v) = v'$ and $v'[0] = 1$ if, and only if, there exists encodings M_v of v and $M_{v'}$ of v', with $M_v \xrightarrow{*} M_{v'}$. Moreover, if there is a marking M such that $M_v \xrightarrow{*} M$ and $0 \in frac(M)$, where M contains a token of age 0, then $M \leq M_{v'}$ for some encoding $M_{v'}$ of a vector $v' = F^m(v)$. This means that it suffices to add one transition t with $Pre(t) = (True_0, 0)$ whose enabledness witnesses the existence of a reachable encoding $M_{v'}$ containing a token $(True_0, 0)$. By the properties above, there exists $m \in \mathbb{N}$ with $F^m(v) = v'$ and $v'[0] = 1$ iff $M_v \xrightarrow{*} M_{v'} \xrightarrow{t}$. □

This lower bound holds even for discrete time TPN, e.g. [37], because the proof uses only timed steps with duration $d = 1$.

2.7 Upper Bound

We will first present a summary of our techniques that we use in order to show the upper bound for ∃COVER for TPNs. Then, we present our formal construction along with explanations and intuitions throughout.

2.7.1 Summary

Without loss of generality, we can restrict ourselves to a TPN which is *non-consuming* (Definition 2.5.1). Intuitively, since we start with an arbitrarily high number of tokens anyway, it does not matter how many tokens are consumed by transitions during the computation, since some of them will always remain. This is illustrated in Lemma 2.7.1, which justifies a logarithmic space reduction from the existential coverability problem for a TPN to the existential coverability problem for a non-consuming TPN. In other words, if one can solve the existential coverability problem for a non-consuming TPN, then one can also solve the existential coverability for a (general) TPN.

In Section 2.7.3, we perform an *abstraction* of the real-valued clocks, similar to the one used in [8]. In other words, clock values are split into integer parts and fractional parts, as shown in Definition 2.7.1. The integer parts of the clocks can be abstracted into a finite domain, since the transition guards cannot distinguish between values above the maximal constant that appears in the system. The fractional parts of the clock values that occur in a marking are ordered sequentially. Hence, every marking can be abstracted into a *string* where all the tokens with the i-th fractional clock value are encoded in the i-th symbol in the string. Since token multiplicities do not matter for existential coverability, the alphabet from which these strings are built is finite.

In order to solve existential coverability (for non-consuming TPN), in Section 2.7.4 we come up with an acceleration procedure (Algorithm 1) that computes a symbolic representation of the set of reachable strings (i.e., marking abstractions), in terms of finitely many regular expressions. We use regular expressions instead of strings only due to the fact that by firing discrete and timed transitions, the space of these strings is still infinite.

Furthermore, to justify the termination of Algorithm 1 (Lemma 2.7.12), we show in Lemma 2.7.11 that the length of the regular expressions that are present in the procedure cannot be more than 4. This is achieved by producing a sequence of transitions via alternating between discrete and timed steps, along with the claim that some symbols can be discarded. Note that without this property, termination of the procedure would

not have been possible. Correctness of Algorithm 1 is shown in Lemma 2.7.13.

Finally, we can check existential coverability for a non-consuming TPN by using this symbolic representation, as in Section 2.7.5. Due to Corollary 2.7.15, the existential coverability problem for TPNs is PSPACE-complete.

2.7.2 Non-consuming TPNs

We show in Lemma 2.7.1 that in terms of existential coverability, one can focus on a non-consuming TPN, i.e., no tokens are consumed during transition firing. More specifically, one can reduce the existential coverability problem for general TPNs to the existential coverability problem for non-consuming TPNs. This is realized using Claim 2.7.1.1.

Lemma 2.7.1. The ∃COVER problem for TPN logspace-reduces to the ∃COVER problem for non-consuming TPN. That is, for every TPN $\mathcal{N}$ and for every place p and transition t of $\mathcal{N}$, one can construct, using logarithmic space, a non-consumimg TPN $\mathcal{N}'$ together with a place p' and transition t' of $\mathcal{N}'$, so that there exists $M \in Cover_{\mathcal{N}}(\mathbb{N} \cdot \{(p,0)\})$ enabling t in $\mathcal{N}$ if and only if there exists $M' \in Cover_{\mathcal{N}'}(\mathbb{N} \cdot \{(p',0)\})$ that enables t' in $\mathcal{N}'$.

Proof. First notice that the first condition in Definition 2.5.1, that asks that every transition takes at most one token each place, is merely a syntactic convenience. A net satisfying this condition can be constructed by adding a few extra places and intermediate transitions to first distribute tokens to those extra places for the original transition to consume.

So let's assume w.l.o.g., that $\mathcal{N}$ satisfies this condition and let $\mathcal{N}'$ be the non-consuming variant derived from $\mathcal{N}$ where for all transitions T, $Post_{\mathcal{N}'}(t) \stackrel{\text{def}}{=} Post_{\mathcal{N}}(t) \oplus Pre_{\mathcal{N}}(t)$. Notice that then, for every discrete step $M \longrightarrow_t M'$ we have that $M \leq M'$. We prove the following claim.

Claim 2.7.1.1. *For every place p and transition t of $\mathcal{N}$ there exists $M \in Cover_{\mathcal{N}}(\mathbb{N} \cdot \{(p,0)\})$ enabling t in $\mathcal{N}$ if, and only if there exists $M' \in Cover_{\mathcal{N}'}(\mathbb{N} \cdot \{(p,0)\})$ that enables t in $\mathcal{N}'$.*

The "$\mathcal{N} \to \mathcal{N}'$" direction follows from the observation that the pointwise ordering $\leq$ on markings, is a simulation: If $M \longrightarrow N$ and $M' \geq M$ then there exists an $N' \geq N$ with $M' \longrightarrow N'$. For the other direction, suppose there exists a witnessing path

$$m \cdot \{(p,0)\} \;=\; M_0 \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \cdots \longrightarrow M_k \xrightarrow{t}$$

of length k in $\mathcal{N}'$. We can inductively derive a witnessing path in $\mathcal{N}$ backwards, again using the fact that $\leq$ is a simulation. First note that if M' enables t, then every $m' \cdot M'$ with $m' > 0$ enables t, (in both nets). Suppose $M_i \xrightarrow{\rho}$ is a path of length $(k-i)$ that ends in a t-transition. By the simulation property, there is such a path from every $m \cdot M_i$, $m > 0$. Further, there must exist markings $M'_{i-1} \in \downarrow(\mathbb{N} \cdot M_{i-1})$ and $M'_i \in \downarrow(\mathbb{N} \cdot M_i)$ such that $M'_{i-1} \longrightarrow M'_i$. It suffices to pick $M'_{i-1} \stackrel{\text{def}}{=} B \cdot M_{i-1}$, where $B \in \mathbb{N}$ is the maximal cardinality of any multiset $Pre(t)$ (This number is itself bounded by $|P| \cdot |Var|$ by our assumption on $Pre(t)$). We conclude that in $\mathcal{N}$ there is a path ending in a t-transition and starting in marking $(B \cdot k) \cdot M_0$, which is in $\mathbb{N} \cdot \{(p,0)\}$. □

2.7.3 Region Abstraction

We recall a constraint system called regions defined for timed automata [12]. The version for TPN used here is similar to the one in [8].

Consider a fixed, nonconsuming TPN $\mathcal{N} = (P, T, Var, G, Pre, Post)$. In order to state formally what an abstraction of a marking is (Definition 2.7.1), we split the clock values of tokens into an *integral* part and a fractional part. Note that since the transition guard constraint system cannot distinguish between clock values above the maximal constant that appears in $\mathcal{N}$, the integer parts can be abstracted into a finite domain, i.e., 0 up to a maximal integer value that appears in the transition guards G, called c_{max}. We order the fractional parts of clock values of tokens in a sequential order. We recall that a marking is an assignment of tokens to places of a TPN. In the non-consuming TPN framework, we interpret markings as sets (since token multiplicities do not matter). Due to the fact that every token has a clock value, an abstraction of a marking consists of a string whose i-th symbol is made of a place p and the integral part of a clock value, whose token is present on p and have i-th fractional part. An example is given in Example 2.7.2. To be more specific, an abstraction of a marking is considered as the *shortest* S-abstraction, where an S-abstraction is made of a superset of all fractional parts of clock values.

More formally, let c_{max} be the largest finite value appearing in transition guards G. Since different tokens with age $> c_{max}$ cannot be distinguished by transition guards, we consider only token ages below or equal to c_{max} and treat the integer parts of older tokens as equal to $c_{max}+1$. Let $int(c) \stackrel{\text{def}}{=} \min\{c_{max}+1, \lfloor c \rfloor\}$ and $frac(c) \stackrel{\text{def}}{=} c - \lfloor c \rfloor$ for a real value $c \in \mathbb{R}_{\geq 0}$. We will work with an abstraction of TPN markings as words over the alphabet $\Sigma \stackrel{\text{def}}{=} 2^{P \times [c_{max}+1]}$. Each symbol $X \in \Sigma$ represents the places and integer ages of tokens for a particular fractional value.

Definition 2.7.1. Let $M \subseteq P \times \mathbb{R}_{\geq 0}$ be a marking and let $frac(M) \stackrel{\text{def}}{=} \{frac(c) \mid (p,c) \in M\}$ be the set of fractional clock values that appear in M.

Let $S \subset [0,1[$ be a finite set of real numbers with $0 \in S$ and $frac(M) \subseteq S$ and let $f_0, f_1, \ldots, f_n$, be an enumeration of S so that $f_{i-1} < f_i$ for all $i \leq n$. The *S-abstraction* of M is

$$abs_S(M) \stackrel{\text{def}}{=} x_0 x_1 \ldots x_n \in \Sigma^*$$

where $x_i \stackrel{\text{def}}{=} \{(p, int(c)) \mid (p,c) \in M \wedge frac(c) = f_i\}$ for all $i \leq n$. We simply write $abs(M)$ for the shortest abstraction, i.e. with respect to $S = \{0\} \cup frac(M)$.

Example 2.7.2. *The abstraction of marking* $M = \{(p,2.1),(q,2.2),(p,5.1),(q,5.1)\}$ *is* $abs(M) = \emptyset\ \{(p,2),(p,5),(q,5)\}\ \{(q,2)\}$. *The first symbol is* $\emptyset$*, because* M *contains no token with an integer age (i.e., no token whose age has fractional part* 0*). The second and third symbols represent sets of tokens with fractional values* 0.1 *and* 0.2*, respectively.*

Note that clocks with integer values play a special role in the behavior of TPN, because the constants in the transition guards are integers. Thus we always include the fractional part 0 in the set S in Definition 2.7.1.

We use a special kind of regular expressions over Σ to represent coverable sets of TPN markings as follows. Informally, the denotation of a regular expression E over alphabet $\Sigma \stackrel{\text{def}}{=} 2^{P \times [c_{max}+1]}$ is made of the set of markings which are covered (i.e., smaller with respect to $\leq$) by another marking whose abstraction belongs to the language of regular expression E. Definition 2.7.2 will be particularly useful when reasoning about the correctness of the acceleration procedure.

Definition 2.7.2. A regular expression E over Σ represents the downward-closed set of TPN markings covered by one that has an abstraction in the language of E:

$$[\![E]\!] \stackrel{\text{def}}{=} \{N \mid \exists M \exists S.\ M \geq N \wedge abs_S(M) \in \mathcal{L}(E)\}.$$

An expression is *simple* if it is of the form $E = x_0 x_1 \ldots x_k$ where for all $i \leq k$ either $x_i \in \Sigma$ or $x_i = y_i^*$ for some $y_i \in \Sigma$. In the latter case we say that x_i *carries a star*. That is, a simple expression is free of Boolean combinators and uses only concatenation and Kleene star. We will write $\hat{x}_i$ to denote the symbol in Σ at position i: it is x_i if $x_i \in \Sigma$ and y_i otherwise.

Remark 2.7.3. Notice that for all simple expressions α, β so that $|\alpha| > 0$, we have that $[\![\alpha\emptyset\beta]\!] = [\![\alpha\beta]\!]$. However, unless α has length 0 or is of the form $\alpha = \emptyset\alpha'$, we have

$[\![\emptyset\alpha]\!] \neq [\![\alpha]\!]$. This is because a marking M that contains a token (p,c) with $frac(c) = 0$ has the property that all abstractions $abs_S(M) = x_0 \ldots x_k$ of M have $x_0 \neq \emptyset$.

The following lemmas express the effect of TPN transitions at the level of the region abstraction. Lemmas 2.7.5 and 2.7.6 state that maximally firing of discrete transitions (the relation $\xrightarrow{*}_D$) is computable and monotone. Lemmas 2.7.7 and 2.7.8 state how to represent timed-step successor markings.

The main idea of Lemma 2.7.5 is essentially due to the monotonicity of discrete transition firing in TPN and the fact that iteratively firing transitions must saturate due to the nonconsuming semantics. We first prove it only for star-free expressions E in condition 3 (Lemma 2.7.4), and then generalize to all simple expressions by induction.

Informally, Lemma 2.7.4 refers to simple expressions without any starred symbols. For every non-consuming TPN, one can compute 2 functions f and g that carry 3 symbols as arguments in the following way. Function f is used to establish what happens to the zero symbol of the marking abstraction when the discrete transition is fired in the TPN, keeping in mind the current abstraction and producing a new symbol. Function g does exactly the same thing, but refers to symbols that correspond to non-zero fractional parts. Note that the α argument refers to the first symbol of the expression (the one which refers to the zero fractional part), whereas the β argument refers to the rest of the symbols (corresponding to non-zero fractional parts). The third argument refers to the current symbol that is taken into account. Note that by firing discrete transitions, the new symbols that are produced are larger w.r.t. subset ordering to the corresponding previous ones, before firing. Hence, point 1 and 2 refer to this property, whereas point 3 justifies the mechanism of discrete transition steps, i.e., the fact that the newly simple expression produced represent exactly those markings that are obtained by using discrete steps.

Lemma 2.7.4. For every non-consuming TPN $\mathcal{N}$ there are polynomial time computable functions $f : \Sigma \times \Sigma \times \Sigma \to \Sigma$ and $g : \Sigma \times \Sigma \times \Sigma \to \Sigma$ with the following properties.

1. f and g are monotone (w.r.t. subset ordering) in each argument.
2. $f(\alpha,\beta,x) \supseteq x$ and $g(\alpha,\beta,x) \supseteq x$ for all $\alpha,\beta,x \in \Sigma$.
3. For every word $w = x_0x_1 \ldots x_k$ over Σ, $\alpha \stackrel{\text{def}}{=} x_0$ and $\beta \stackrel{\text{def}}{=} \bigcup_{i>0} x_i$, and $w' \stackrel{\text{def}}{=} f(\alpha,\beta,x_0)g(\alpha,\beta,x_1)\ldots g(\alpha,\beta,x_k)$ we have $[\![w']\!] = \{M'' \mid \exists M \in [\![w]\!] \wedge M \xrightarrow{*}_{Disc} M' \geq M''\}$.

Proof. It is sufficient to show the existence of such functions f_t and g_t for individual transitions $t \in T$ and $\longrightarrow_t$ instead of $\xrightarrow{*}_{Disc}$. The functions f and g can then be obtained by iterated applications of f_t and g_t (for all transitions t) until convergence. (In addition to expanding x, the results of each application f_t and g_t are also added to α and β, respectively.) This works, because the functions f_t and g_t are monotone and operate on the finite domain/range Σ. Since we have a polynomial number of transitions, and each symbol in Σ can increase (by strict subset ordering) at most $|P| \cdot (c_{max}+1)$ times, the number of iterations is polynomial. Moreover, the properties of Item 1, Item 2 and Item 3 carry over directly from f_t and g_t to f and g, respectively.

Now we consider the definitions and properties of the functions f_t and g_t for a particular transition t. Given a variable evaluation $\pi : Var \to \mathbb{R}_{\geq 0}$, we define the functions π_0 and $\pi_{>0}$ from sets over $(P \times Var)$ to sets over $(P \times \mathbb{N})$ as follows. Intuitively, they cover the parts of the assignment π with zero/nonzero fractional values, respectively. Let $\pi_0(S) \stackrel{\text{def}}{=} \{(p,c) \mid (p,y) \in S \wedge \pi(y) = c \in \mathbb{N}\}$ and $\pi_{>0}(S) \stackrel{\text{def}}{=} \{(p,c) \mid (p,y) \in S \wedge \lfloor \pi(y) \rfloor = c \wedge frac(\pi(y)) > 0\}$. The definitions are lifted to multisets in the straightforward way.

Now let t be a transition. We say that (α,β) enables t iff $\exists \pi : Var \to \mathbb{R}_{\geq 0}$ such that $\pi(y) \in G(t)(y)$ for all variables y and $\pi_0(Pre(t)) \subseteq \alpha$ and $\pi_{>0}(Pre(t)) \subseteq \beta$. Thus if $abs(M) = x_0 x_1 \ldots x_n$ then M enables t iff $(x_0, \bigcup_{i>0} x_i)$ enables t, since all transition guards in $G(t)$ are intervals bounded by integers (i.e., t cannot distinguish between different nonzero fractional values). Moreover, enabledness can be checked in polynomial time (choose integers for the part in α and rationals with fractional part $1/2$ for the part in β).

In the case where (α,β) does not enable t we just let $g_t(\alpha,\beta,x) \stackrel{\text{def}}{=} x$ and $f_t(\alpha,\beta,x) \stackrel{\text{def}}{=} x$. The conditions above are trivially satisfied in this case.

In the case where (α,β) enables t, let $g_t(\alpha,\beta,x) \stackrel{\text{def}}{=} x \cup \gamma$ where γ is defined as follows. We have $(p,c) \in \gamma$ iff there is a $(p,y) \in Post(t)$ and $(q,y) \in Pre(t)$ such that $(q,c) \in x$. Similarly, let $f_t(\alpha,\beta,x) \stackrel{\text{def}}{=} x \cup \gamma$ where γ is defined as follows. We have $(p,c) \in \gamma$ iff either (1) there is a $(p,y) \in Post(t)$ and $(q,y) \in Pre(t)$ such that $(q,c) \in x$, or (2) $c = 0$ and there is a $(p,0) \in Post(t)$. All these conditions can be checked in polynomial time. Item 1 and Item 2 follow directly from the definition.

Towards Item 3, we show $[\![w']\!] \supseteq \{M'' \mid \exists M \in [\![w]\!] \wedge M \longrightarrow_t M' \geq M''\}$. (The proof of the reverse inclusion $\subseteq$ is similar.) Let $w = x_0 x_1 \ldots x_k$, $\alpha \stackrel{\text{def}}{=} x_0$, $\beta \stackrel{\text{def}}{=} \bigcup_{i>0} x_i$ such that (α,β) enables t and $w' \stackrel{\text{def}}{=} f_t(\alpha,\beta,x_0) g_t(\alpha,\beta,x_1) \ldots g_t(\alpha,\beta,x_k)$. If $M \in [\![w]\!]$ and $M \longrightarrow_t M'$ then $M' \geq M$ since $\mathcal{N}$ is non-consuming. We show that every additional

token $(p,u) \in M' \ominus M$ is included in $[\![w']\!]$. (This implies the inclusion above, since $M' \ominus M \geq M'' \ominus M$.) For every additional token $(p,u) \in M' \ominus M$ there are two cases.

- Assume $frac(u) > 0$. Then the token (p,u) must have inherited its clock value from some token $(q,u) \in M$ via a variable y specified in the Pre/Post of t (since discrete transitions cannot create new fractional parts of clock values). This case is covered by γ in the definition of g_t above. In particular, if $(q,u) \in M$ was abstracted to x_i in w then $(p,u) \in M'$ is abstracted to $g_t(\alpha,\beta,x_i)$ in w'.

- Assume $frac(u) = 0$. Then there are two cases. In the first case the token (p,u) inherited its clock value from some token $(q,u) \in M$ via a variable y specified in the Pre/Post of t. This case is covered by part (1) of γ in the definition of f_t above. In particular, $(q,u) \in M$ was abstracted to x_0 in w, because $frac(u) = 0$. Thus $(p,u) \in M'$ is abstracted to $f_t(\alpha,\beta,x_0)$ in w'. In the second case the token (p,u) got its clock value via a clock-reset to zero. This case is covered by part (2) of γ in the definition of f_t above. In particular, in this case we must have $u = 0$, and $(p,0) \in M'$ was abstracted to $f_t(\alpha,\beta,x_0)$ in w'.

It follows that $abs(M') \leq w'$, i.e., by the ordering on symbols in Σ, every letter in $abs(M')$ is smaller than the corresponding letter in w'. Thus $M' \in [\![w']\!]$. Since $M' \geq M''$ and $[\![w']\!]$ is downward closed, we also have $M'' \in [\![w']\!]$ as required. □

The following lemma (Lemma 2.7.5) extends the properties 1, 2, and 3 of Lemma 2.7.5 to simple expressions (i.e., regular expressions with symbols that are either starred or without a star). The intuition of point 3 of Lemma 2.7.5 is very similar to the one in Lemma 2.7.4. First, we can treat every symbol of a simple expression in terms of being a correspondent to the zero fractional part or not. Secondly, the function f takes information about the structure of the previous expression and outputs a new symbol that corresponds to the zero fractional part. The same argument applies for function g, which refers to the remaining symbols (corresponding to non-zero fractional parts).

Therefore, we generalize this property to simple expressions by induction.

Lemma 2.7.5. For every non-consuming TPN $\mathcal{N}$ there are polynomial time computable functions $f : \Sigma \times \Sigma \times \Sigma \to \Sigma$ and $g : \Sigma \times \Sigma \times \Sigma \to \Sigma$ with the following properties.

1. f and g are monotone (w.r.t. subset ordering) in each argument.

2. $f(\alpha,\beta,x) \supseteq x$ and $g(\alpha,\beta,x) \supseteq x$ for all $\alpha,\beta,x \in \Sigma$.

3. Suppose that $E = x_0 x_1 \dots x_k$ is a simple expression, $\alpha \stackrel{\text{def}}{=} x_0$ and $\beta \stackrel{\text{def}}{=} \bigcup_{i>0} \hat{x}_i$, and $E' = x'_0 x'_1 \dots x'_k$ is the derived expression defined by conditions:

   (a) $x'_0 \stackrel{\text{def}}{=} f(\alpha, \beta, x_0)$,

   (b) $x'_i \stackrel{\text{def}}{=} g(\alpha, \beta, \hat{x}_i)^*$ for $i > 0$,

   (c) x'_i carries a star iff x_i does.

   Then $[\![E']\!] = \{M'' \mid \exists M \in [\![E]\!] \wedge M \xrightarrow{*}_{Disc} M' \geq M''\}$.

Proof. Let f and g be the functions from Lemma 2.7.4, which immediately yields Item 1 and Item 2. Towards Item 3, consider all words w in $\mathcal{L}(E)$ that contain each starred symbol in E at least once. (The other cases are irrelevant for $[\![E]\!]$ since they are subsumed by monotonicity.) For each such word w, the α, β derived from w in Lemma 2.7.4 are the same as the α, β derived from E in Item 3. If x_i in E carries a star then w contains a corresponding nonempty subsequence $x_i \dots x_i$. We apply Lemma 2.7.4 to each such w to obtain the corresponding w'. The word w' then contains the corresponding subsequence $g(\alpha, \beta, x_i) \dots g(\alpha, \beta, x_i)$. Let E' then be defined as in Item 3, i.e., by applying functions to the symbols and keeping the stars at the same symbols as in E. By Lemma 2.7.4, this is computable in polynomial time. We have $\mathcal{L}(E') = \bigcup_{w \in \mathcal{L}(E)} \{w'\}$. Thus $[\![E']\!] = \bigcup_{w \in \mathcal{L}(E)} [\![w']\!] = \bigcup_{w \in \mathcal{L}(E)} \{M'' \mid \exists M \in [\![w]\!] \wedge M \xrightarrow{*}_{Disc} M' \geq M''\} = \{M'' \mid \exists M \in [\![E]\!] \wedge M \xrightarrow{*}_{Disc} M' \geq M''\}$ for Item 3 as required. □

In Definition 2.7.3, for a given expression E, we define the notion of successor expression obtained by firing discrete transition steps, denoted as $SAT(E)$.

Definition 2.7.3. We will write $SAT(E) \stackrel{\text{def}}{=} E'$ for the successor expression E' of E guaranteed by Lemma 2.7.5. I.e., $SAT(E)$ is the saturation of E by maximally firing discrete transitions.

Notice that by definition it holds that $[\![E]\!] \subseteq [\![SAT(E)]\!] \subseteq Cover([\![E]\!])$, and consequently also that $Cover([\![SAT(E)]\!]) = Cover([\![E]\!])$.

The following lemma establishes a connection between two expresssions X and Y and their corresponding successor expressions. Namely, if a symbol on position i in X is smaller w.r.t. subset ordering than a symbol on position i in Y, then the corresponding symbols in the successor expressions preserve that ordering.

Lemma 2.7.6. Suppose that $X = x_0x_1 \dots x_k$ is a simple expression of length $k+1$ with $SAT(X) = x'_0x'_1 \dots x'_k$ and $x_0, x'_0 \in \Sigma$. Let $Y = y_0\alpha_1y_1\alpha_2 \dots \alpha_ky_k$ be a simple expression with $SAT(Y) = y'_0\alpha'_1y'_1\alpha'_2 \dots \alpha'_ky'_k$ and $y_0, y'_0 \in \Sigma$.

If $\hat{x}_i \subseteq \hat{y}_i$ for all $i \leq k$ then $\hat{x}'_i \subseteq \hat{y}'_i$ for all $i \leq k$.

Proof. The assumption of the lemma provides that $\alpha_x \stackrel{\text{def}}{=} x_0 \subseteq \alpha_y \stackrel{\text{def}}{=} y_0$ and $\beta_x \stackrel{\text{def}}{=} \bigcup_{k \geq i > 0} \hat{x}_i \subseteq \beta_y \stackrel{\text{def}}{=} \bigcup_{k \geq i > 0} \hat{y}_i$. Therefore, by Item 1 of Lemma 2.7.5, we get that

$$x'_0 = f(\alpha_x, \beta_x, x_0) \quad \subseteq \quad f(\alpha_y, \beta_y, y_0) = y'_0$$

and similarly, for all $k \geq i \geq 0$, that $\hat{x}'_i = g(\alpha_x, \beta_x, \hat{x}_i) \subseteq g(\alpha_y, \beta_y, \hat{y}_i) = \hat{y}'_i$. □

Lemma 2.7.7 and Lemma 2.7.8 justify the timed steps as presented in Definition 2.3.4. Namely, Lemma 2.7.7 refers to the case when the time that passes does not go over the maximal fractional part of clock values. In this case, the new fractional parts are incremented by a sufficiently small amount such that the integer parts remain intact, along with the symbol that corresponds to the zero fractional part being empty. However, Lemma 2.7.8 justifies the case when the time that passes goes over the maximal fractional part of clock values. In this case, some integer parts are incremented by 1.

For $x \in \Sigma$ we write $(x+1) \stackrel{\text{def}}{=} \{(p, int(n+1)) \mid (p,n) \in x\}$ for the symbol where token ages are incremented by 1.

Lemma 2.7.7. $[\![\emptyset E]\!] = \{M' \mid \exists M \in [\![E]\!] \wedge M \longrightarrow_d M' \wedge d < 1 - \max(frac(M))\}$.

Proof. "$\supseteq$": Suppose that M is a non-empty marking in $[\![E]\!]$, $d < 1 - \max(frac(M))$ and $M \longrightarrow_d M'$. The assumption on d implies that for every token $(p,c) \in M$ we have $int(c) = int(c+d)$. In other words, the integral part of the token age remained the same. Therefore $(p, int(c)) = (p, int(c+d)) \in M'$. Also from the assumption on d we get that

$$frac(M') = \{x + d \mid x \in frac(M)\}$$

Recall that $abs(M) = abs_S(M)$ and $abs(M') = abs_{S'}(M')$ for the sets $S \stackrel{\text{def}}{=} \{0\} \cup frac(M)$ and $S' \stackrel{\text{def}}{=} \{0\} \cup frac(M')$. Clearly, $0 \notin frac(M')$. There are two cases:

1. $0 \in frac(M)$. Then $abs(M') = \emptyset abs(M) \in \mathcal{L}(\emptyset E)$, and consequently, $M' \in [\![\emptyset E]\!]$.

2. $0 \notin frac(M)$. Then $abs(M') = abs(M) = \emptyset w \in \mathcal{L}(E)$. Suppose that $E = x_0\alpha$, i.e., E has $x_0 \in \Sigma$ as its leftmost symbol, and $w \in \mathcal{L}(\alpha)$. If $x_0 = \emptyset$ then $[\![E]\!] = [\![\emptyset E]\!]$ and thus $abs(M') \in [\![\emptyset E]\!]$. Otherwise, if $x_0 \neq \emptyset$ then $x_0 w \in \mathcal{L}(E)$ and $x_0 w = abs(M'')$ for some marking $M'' \geq M'$. So again, $M' \in [\![\emptyset E]\!]$.

"$\subseteq$": W.l.o.g., pick a non-empty marking $M' \in [\![\emptyset E]\!]$. If E has $\emptyset$ as its leftmost symbol, then $[\![\emptyset E]\!] = [\![E]\!]$ and the claim follows using $d = 0$, since then $M' \in [\![E]\!]$. So suppose that E does not start with $\emptyset$. Note that by Definition 2.7.1, there are no tokens in the marking M' whose clocks have fractional value zero. Let

$$d \stackrel{\text{def}}{=} \min(frac(M'))$$

be the minimal fractional clock value among the tokens of M' and based on this, define $M \stackrel{\text{def}}{=} \{(p, c-d) \mid (p,c) \in N'\}$. By construction of M we get $M \longrightarrow_d M'$ and also that $\max(frac(M)) = \max(frac(M')) - d < 1$. Therefore that $1 - \max(frac(M)) < 1 - d$. Finally, observe that $frac(M) = \{x - d \mid x \in frac(M')\}$ and $0 \in frac(M)$. It follows that $abs(M') = \emptyset abs(M)$ and therefore that $abs(M) \in \mathcal{L}(E)$ and $M \in [\![E]\!]$. This means that M' is included in the set on the right in the claim. □

Lemma 2.7.8. Let αz be a simple expression where $\hat{z} = z \in \Sigma$ (the rightmost symbol is not starred). Then, $[\![(z+1)\alpha]\!]$ contains a marking N if, and only if, there exists markings $N' \geq N$ and M, and a set $S \subseteq [0,1[$ so that

1. $|S| = |\alpha z|$
2. $abs_S(M) \in \mathcal{L}(\alpha z)$
3. $M \longrightarrow_d N'$ for $d = 1 - \max(S)$.

Proof. Suppose markings N, N', M, a set $S \subseteq [0,1[$ and $d \in \mathbb{R}_{\geq 0}$ so that the conditions 1 to 3 are satisfied. Let $S' \stackrel{\text{def}}{=} \{0\} \cup \{s + d \mid s \in S \setminus \{d\}\}$. Then, $|S'| = |S|$ and $abs_{S'}(N') \in \mathcal{L}((z+1)\alpha)$, which witnesses that $N \in [\![(z+1)\alpha]\!]$.

Conversely, let $N \in [\![(z+1)\alpha]\!]$ be a non-empty marking. If $|\alpha| = 0$, then $N \in [\![(z+1)]\!]$ and so $abs_S(N) \in \mathcal{L}((z+1))$ for $S \stackrel{\text{def}}{=} frac(N) = \{0\}$. This means that $M \longrightarrow_1 N = (M+1)$ for a marking M with $abs_S(M) \in \mathcal{L}(z) = \mathcal{L}(\alpha z)$.

If $|\alpha| > 0$, pick some marking $N' \geq N$ and set S' so that $abs_{S'}(N') = (z+1)w$, for some word $w \in \mathcal{L}(\alpha)$. Then we must have that $|S'| = |(z+1)\alpha| > 1$ and so $d \stackrel{\text{def}}{=} \min(S' \setminus \{0\})$ exists. Let $S \stackrel{\text{def}}{=} \{s - d \mid s \in S'\} \cup \{1-d\}$ and M be the unique marking with $M \longrightarrow_d N'$. Notice that $1 - d = \max(S)$. It follows that $abs_S(M) = wz \in \mathcal{L}(\alpha z)$. □

We will often use the following simple fact, which is a direct consequence of Lemma 2.7.8.

Corollary 2.7.9. $[\![(z+1)\alpha]\!] \subseteq Cover([\![\alpha z]\!])$.

Finally, the following lemma will be the basis for our exploration algorithm.

Lemma 2.7.10. Let αx_0^* be a simple expression with $SAT(\alpha x_0^*) = \alpha x_0^*$. Then $Cover\left([\![\alpha x_0^*]\!]\right) = [\![\alpha x_0^*]\!] \cup Cover\left([\![(x_0+1)\alpha x_0^*]\!]\right)$.

Proof. For the right to left inclusion notice that $[\![\alpha x_0^*]\!] \subseteq Cover\left([\![\alpha x_0^*]\!]\right)$ trivially holds. For the rest, we have $[\![(x_0+1)\alpha x_0^*]\!] \subseteq Cover\left([\![\alpha x_0^*]\!]\right)$ by Corollary 2.7.9, and therefore $Cover\left([\![(x_0+1)\alpha x_0^*]\!]\right) \subseteq Cover\left(Cover\left([\![\alpha x_0^*]\!]\right)\right) = Cover\left([\![\alpha x_0^*]\!]\right)$. For the left to right inclusion, we equivalently show that

$$Cover\left([\![\alpha x_0^*]\!]\right) \setminus [\![\alpha x_0^*]\!] \subseteq Cover\left([\![(x_0+1)\alpha x_0^*]\!]\right) \tag{2.1}$$

Using the assumption that $SAT(\alpha x_0^*) = \alpha x_0^*$, the set on the left contains everything coverable from $[\![\alpha x_0^*]\!]$ by a sequence that starts with a (short) time step. It can therefore be written as

$$Cover\left(\{N_1 \mid \exists N_0 \in [\![\alpha x_0^*]\!] \wedge N_0 \longrightarrow_d N_1 \wedge 0 < d < 1 - \max(frac(N_0))\}\right).$$

By Lemma 2.7.7 and because $[\![\emptyset\alpha]\!] \subseteq [\![X\alpha]\!]$ for all $X \in \Sigma$ and $\alpha \in \Sigma^*$, we conclude that indeed, $Cover\left([\![\alpha x_0^*]\!]\right) \setminus [\![\alpha x_0^*]\!] \subseteq Cover\left([\![\emptyset\alpha x_0^*]\!]\right) \subseteq Cover\left([\![(x_0+1)\alpha x_0^*]\!]\right)$. □

2.7.4 Acceleration

We propose an acceleration procedure based on unfolding expressions according to Lemma 2.7.10 (interleaved with saturation steps to guarantee its premise) and introducing new Kleene stars to keep the length of intermediate expressions bounded. This procedure (depicted in Algorithm 1), is used to characterize an initial subset of the coverability set. Note that the length of the regular expressions that are present in the procedure cannot be more than 4, due to one crucial aspect, namely Lemma 2.7.11, where some symbols can be discarded because they are redundant (i.e., smaller w.r.t. subset ordering).

We now present the actual acceleration procedure.

Algorithm 1 Accelerate

Input: a simple expression $S_0 = x_1 x_0^*$ (of length 2 and with last symbol starred)

Output: simple expressions S_1, S_i and R, of lengths 2, 4, and 2, respectively.

1: $S_1 \stackrel{\text{def}}{=} x_1^1 (x_0^1)^* = SAT(x_1 x_0^*)$

2: $S_2 \stackrel{\text{def}}{=} x_2^2 x_1^2 (x_0^2)^* = SAT((x_0^1 + 1) S_1)$

3: $S_3 \stackrel{\text{def}}{=} x_3^3 x_2^3 x_1^3 (x_0^3)^* = SAT((x_0^2 + 1) S_2)$

4: $i \leftarrow 3$

5: **repeat**

6: $\quad x_{i+1}^{i+1} x_i^{i+1} x_{i-1}^{i+1} x_1^{i+1} (x_0^{i+1})^* \stackrel{\text{def}}{=} SAT((x_0^i + 1) S_i)$

7: $\quad S_{i+1} \stackrel{\text{def}}{=} x_{i+1}^{i+1} (x_i^{i+1})^* x_1^{i+1} (x_0^{i+1})^*$

8: $\quad i \leftarrow i + 1$

9: **until** $S_i = S_{i-1}$

10: $R \stackrel{\text{def}}{=} (x_1^i + 1)(x_{i-1}^i)^*$

11: **Return** S_1, S_i, R

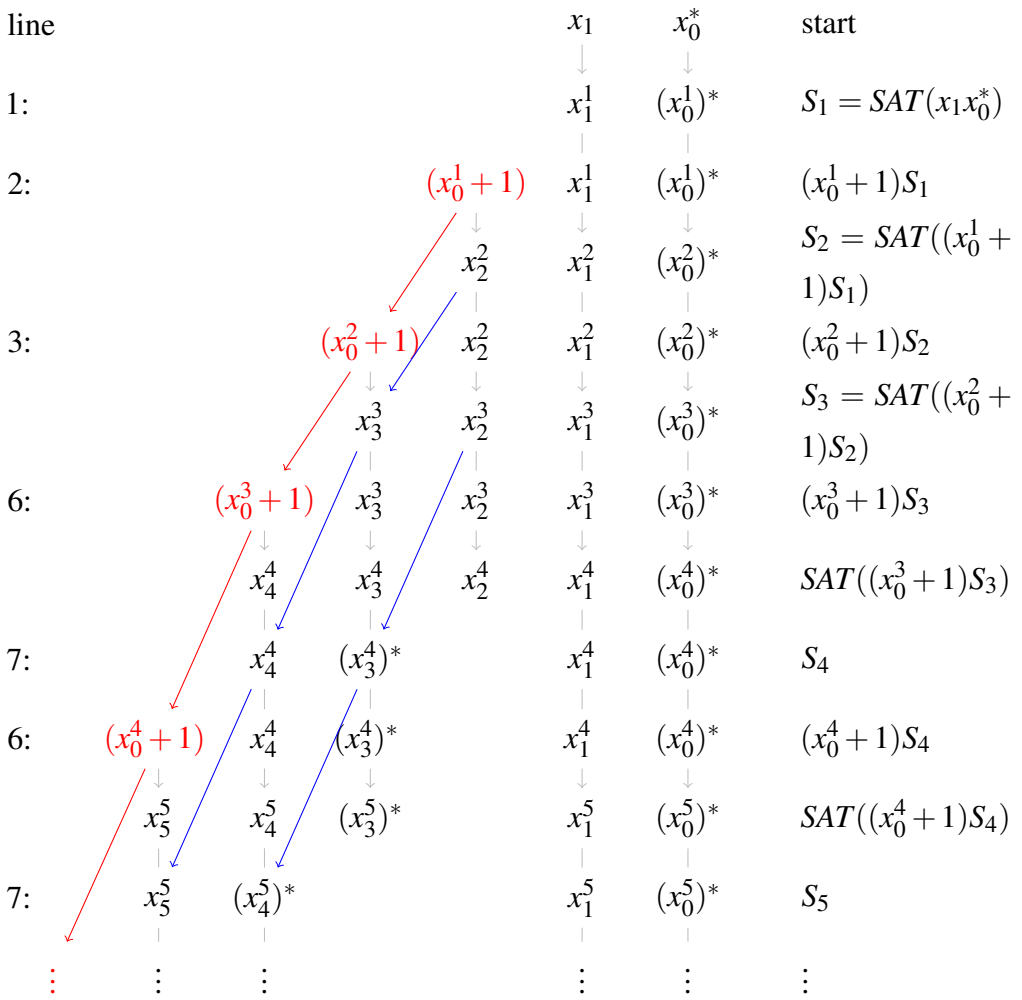


Figure 2.2: A run of Algorithm 1 (initial steps). The column on the left indicates the line of code, the middle depicts the current expression and the column on the right recalls its origin. Gray bars indicate that the respective symbols are equal. Arrows denote (set) inclusion between symbols. The gray vertical arrows indicate inclusions due to saturation (Lemma 2.7.5), as claimed in item 1 of Lemma 2.7.11. Red and blue arrows indicate derived inclusions (as stated in Lemma 2.7.11).

Intuitively, given a length-2 simple expression S_0 where the rightmost symbol is starred, the algorithm will first saturate (Definition 2.7.3, in line 1), and then alternatingly rotate a copy of the rightmost symbol (Lemma 2.7.8), and saturate the result (see lines 2, 3, 6). Since each such round extends the length of the expression by one, we additionally collapse them (in line 7) by adding an extra Kleene star to the symbol at the second position. The crucial observation for the correctness of this procedure is that the subsumption step in line 7 does not change the cover sets of the respective expressions.

Observe that Algorithm 1 is well defined because the $SAT(S_i)$ are computable by Lemma 2.7.5. Termination is guaranteed by the following simple observation, which is also the reason that the length of the simple expressions produced by this acceleration procedure cannot have length more than 4.

Lemma 2.7.11. Let $x_j^i \in \Sigma$ be the symbols computed by Algorithm 1. Then

1. $x_j^{i+1} \supseteq x_j^i$, for all $i > j \geq 0$.
2. $x_i^i \supseteq x_{i-1}^{i-1}$ and $x_i^{i+1} \supseteq x_{i-1}^i$, for all $i \geq 3$.

Proof. The first item is guaranteed by Point 2 of Lemma 2.7.5. In particular this means that $x_0^{i+1} \supseteq x_0^i$ and therefore that $(x_0^{i+1}+1) \supseteq (x_0^i+1)$ for all $i \geq 0$ (indicated as red arrows in Figure 2.2). The second item now follows from this observation by Lemma 2.7.6. □

Lemma 2.7.12 (Termination)**.** Algorithm 1 terminates with $i \leq 4 \cdot |P| \cdot (c_{max}+1)$.

Proof. From Lemma 2.7.11 we deduce that for all $i \geq 2$, the expression S_{i+1} is pointwise larger than or equal to S_i with respect to the subset ordering on symbols. The claim now follows from the observation that all expressions $S_{i\geq 3}$ have length 4 and that every symbol $x_i \in \Sigma$ can only increase at most $|P| \cdot (c_{max}+1)$ times. □

We now prove the correctness of Algorithm 1. We show that the cover set of the initial expression $x_1 x_0^*$ is made of 3 parts - first, it is made of the successor expression of $x_1 x_0^*$ by firing discrete transition steps, along with the expression obtained in the fixpoint iteration along with the expression obtained by dropping the last (starred) symbol and incrementing the first symbol by 1.

Lemma 2.7.13 (Correctness)**.** Suppose that S_1, S_ℓ, R be the expressions computed by Algorithm 1 applied to the simple expression $x_1 x_0^*$. Then $Cover\left([\![x_1 x_0^*]\!]\right) = [\![S_1]\!] \cup [\![S_\ell]\!] \cup Cover\left([\![R]\!]\right)$.

Proof. Let $S_1, \ldots S_\ell$ denote the expressions defined in lines 1,2,3, and 7 of the algorithm. That is, ℓ is the least index i such that $S_{i+1} = S_i$. We define a sequence E_i of expressions inductively, starting with $E_1 \stackrel{\text{def}}{=} S_1$ and if $E_i = e_i^i e_{i-1}^i \ldots e_0^i$, we let $E_{i+1} \stackrel{\text{def}}{=} e_{i+1}^{i+1} e_i^{i+1} e_{i-1}^{i+1} \ldots e_0^{i+1} \stackrel{\text{def}}{=} SAT((\hat{e}_0^i+1)E_i)$. Here, the superscript indicates the position of a symbol and not iteration. This is the sequence of expressions resulting from unfolding Lemma 2.7.10, interleaved with saturation steps, just in line 6 of the algorithm. That is, the expressions E_i are *not* collapsed (line 7) and instead grow in

length with i. Still, $E_1 = S_1$, $E_2 = S_2$ and $E_2 = S_3$, but $E_4 \neq S_4$, because the latter is the result of applying the subsumption step of line 7 in our algorithm. Notice that $\mathit{Cover}\left(\llbracket x_1 x_0^* \rrbracket\right) = \left(\bigcup_{k-1 \geq i \geq 1} \llbracket E_i \rrbracket\right) \cup \mathit{Cover}\left(\llbracket E_k \rrbracket\right)$ holds for all $k \in \mathbb{N}$. We will use that

$$\bigcup_{i \geq 2} \llbracket E_i \rrbracket = \bigcup_{i \geq 2} \llbracket S_i \rrbracket = \llbracket S_\ell \rrbracket. \tag{2.2}$$

We start by observing that for all $i, j \in \mathbb{N}$ it holds that $e_j^i = x_j^i$. For $i \leq 3$ this holds trivially by definition of $E_i = S_i$. For larger i, this can be seen by induction using Lemma 2.7.5. Towards the first equality in Equation (2.2), let S_i^j be the expression resulting from $S_i = x_i^i (x_{i-1}^i)^* x_1^i (x_0^i)^*$ by unfolding the first star j times. That is, $S_i^j \stackrel{\text{def}}{=} x_i^i (x_{i-1}^i)^{(j)} x_1^i (x_0^i)^*$, where the superscript (j) denotes j-fold concatenation. Clearly, $\llbracket S_i \rrbracket = \bigcup_{j \geq 0} \llbracket S_i^j \rrbracket$ and so the $\supseteq$-direction of the first equality in Equation (2.2) follows by

$$\begin{aligned}
\llbracket S_i^j \rrbracket = \llbracket x_i^i (x_{i-1}^i)^{(j)} x_1^i (x_0^i)^* \rrbracket &\subseteq \llbracket x_{i+j}^{i+j} \left(x_{i+j-1}^{i+j} x_{i+j-2}^{i+j} \ldots x_i^{i+j} \right) x_1^{i+1} (x_0^{i+1})^* \rrbracket \\
&\subseteq \llbracket x_{i+j}^{i+j} \left(x_{i+j-1}^{i+j} x_{i+j-2}^{i+j} \ldots x_i^{i+j} \right) \left(x_{i-1}^{i+j} \ldots x_2^{i+j} \right) x_1^{i+1} (x_0^{i+j})^* \rrbracket \\
&= \llbracket E_{i+j} \rrbracket,
\end{aligned}$$

where the first inclusion is due to Lemma 2.7.11. The same helps for the other direction:

$$\llbracket E_i \rrbracket = \llbracket x_i^i x_{i-1}^i x_{i-2}^i \ldots x_2^i x_1^i x_0^i \rrbracket \subseteq \llbracket x_i^i (x_{i-1}^i)^{(i-2)} x_1^i x_0^i \rrbracket = \llbracket S_i^{i-2} \rrbracket = \llbracket S_i \rrbracket, \tag{2.3}$$

which completes the proof of the first equality in Equation (2.2). The second equality holds because $\llbracket S_i \rrbracket \subseteq \llbracket S_{i+1} \rrbracket$ for all $i \geq 2$, by Lemma 2.7.11, and by definition of $S_\ell = S_{\ell+1}$. As a next step we show that

$$\mathit{Cover}\left(\llbracket S_\ell \rrbracket\right) = \llbracket S_\ell \rrbracket \cup \mathit{Cover}\left(\llbracket R \rrbracket\right) \tag{2.4}$$

First observe that $\llbracket R \rrbracket = \llbracket (x_1^\ell + 1)(x_{\ell-1}^\ell)^* \rrbracket = \llbracket (x_1^\ell + 1) x_\ell^\ell (x_{\ell-1}^\ell)^* \rrbracket$ and consequently,

$$\begin{aligned}
\mathit{Cover}\left(\llbracket R \rrbracket\right) &= \mathit{Cover}\left(\llbracket (x_1^\ell + 1) x_\ell^\ell (x_{\ell-1}^\ell)^* \rrbracket \right) \\
&\subseteq \mathit{Cover}\left(\llbracket x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell \rrbracket \right) \\
&\subseteq \mathit{Cover}\left(\llbracket x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell (x_0^\ell)^* \rrbracket \right) = \mathit{Cover}\left(\llbracket S_\ell \rrbracket\right)
\end{aligned}$$

where the first equation follows by Corollary 2.7.9 and the second because $\mathcal{L}\left(x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell \right) \subseteq \mathcal{L}\left(x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell (x_0^\ell)^* \right)$. For the left to right inclusion in Equation (2.4), consider a marking $M \in \mathit{Cover}\left(\llbracket S_\ell \rrbracket\right) \setminus \llbracket S_\ell \rrbracket$. We show that $M \in \mathit{Cover}\left(\llbracket R \rrbracket\right)$. Recall that $\mathit{Cover}\left(\llbracket S_\ell \rrbracket\right)$

consists of all those markings M so that there exists a finite path

$$M_0 \xrightarrow{*}_D M'_0 \xrightarrow{d_1}_T M_1 \xrightarrow{*}_D M'_1 \xrightarrow{d_2}_T M_2 \ldots M'_{k-1} \xrightarrow{*}_D M_k$$

alternating between timed and (sequences of) discrete transition steps, with $M_0 \in [\![S_\ell]\!]$, $M_k \geq M$ and all $d_i \leq \max(frac(M'_i))$.

By our choice of M, there must be a first expression in the sequence which is not a member of $[\![S_\ell]\!]$. Since $[\![SAT(S_\ell)]\!] = [\![S_\ell]\!]$, we can assume an index $i > 0$ so that $M_i \notin [\![S_\ell]\!]$ but $M'_{i-1} \in [\![S_\ell]\!]$ that is, the step that takes us out of $[\![S_\ell]\!]$ is a timed step.

Because $[\![S_\ell]\!] = \bigcup_{i\geq 2}[\![S_i]\!]$, it must hold that $M'_{i-1} \in [\![S_j]\!] = [\![x_j^j (x_{j-1}^j)^* x_1^j (x_0^j)^*]\!]$ for some index $j \geq 2$. We claim that it already holds that

$$M'_{i-1} \in [\![x_j^j (x_{j-1}^j)^* x_1^j]\!]. \tag{2.5}$$

Suppose not. If $d_i < \max(frac(M'_{i-1}))$ then $M_i \in [\![\emptyset S_j]\!] \subseteq [\![S_j]\!]$ by Lemma 2.7.7, contradiction. Otherwise, if $d_i = \max(frac(M'_{i-1}))$, notice that every abstraction $abs_S(M'_{i-1}) \in \mathcal{L}(S_j)$ must have $|S| = 4$. So by Lemma 2.7.8, $M_i \in [\![(x_0^j + 1)S_j]\!]$. But then again

$$[\![(x_0^j + 1)S_j]\!] \subseteq [\![SAT((x_0^j + 1)S_j)]\!] \subseteq [\![S_{j+1}]\!], \tag{2.6}$$

contradicting our assumption that $M_i \notin [\![S_\ell]\!]$. Therefore Equation (2.5) holds. By Lemma 2.7.8 we derive that $M_i \in [\![(x_1^j + 1)x_j^j (x_{j-1}^j)^*]\!] = [\![(x_1^j + 1)(x_{j-1}^j)^*]\!] \subseteq [\![(x_1^\ell + 1)(x_{\ell-1}^\ell)^*]\!] = [\![R]\!]$. This concludes the proof of Equation (2.4).

Notice that by Lemma 2.7.10 we have that

$$Cover([\![x_1 x_0^*]\!]) = [\![SAT(x_1 x_0^*)]\!] \cup Cover([\![SAT(x_1 x_0^*)]\!]) = [\![S_1]\!] \cup Cover([\![S_1]\!]). \tag{2.7}$$

Analogously, we get for every $i \geq 1$ that

$$Cover([\![E_i]\!]) = [\![SAT(E_i)]\!] \cup Cover\left([\![SAT((x_0^i + 1)E_i)]\!]\right) = [\![E_i]\!] \cup Cover([\![E_{i+1}]\!]) \tag{2.8}$$

This used Lemma 2.7.10 and the fact that $SAT(E_i) = E_i$ by construction. Using Equation (2.8) and that $[\![E_i]\!] \subseteq [\![E_{i+1}]\!]$ for $i \geq 2$, we deduce

$$Cover([\![S_1]\!]) = Cover([\![E_1]\!]) = [\![E_1]\!] \cup \left(\bigcup_{i\geq 2} Cover([\![E_i]\!])\right). \tag{2.9}$$

Finally we can conclude the desired result as follows.

$$\begin{aligned} Cover([\![x_1 x_0^*]\!]) &\overset{(2.7)}{=} [\![S_1]\!] \cup Cover([\![S_1]\!]) \overset{(2.9)}{=} [\![S_1]\!] \cup Cover\left(\bigcup_{i\geq 2}[\![E_i]\!]\right) \\ &\overset{(2.2)}{=} [\![S_1]\!] \cup Cover([\![S_\ell]\!]) \\ &\overset{(2.4)}{=} [\![S_1]\!] \cup [\![S_\ell]\!] \cup Cover([\![R]\!]) \end{aligned}$$

□

2.7.5 Main Result

The following theorem summarizes our main claims regarding the ∃COVER problem for Timed Petri Nets. Note that all numbers for representing clock values are encoded in unary.

Theorem 2.7.14. *Consider an instance of ∃COVER with $\mathcal{N} = (P, T, Var, G, Pre, Post)$ a non-consuming TPN where c_{max} is the largest constant appearing in the transition guards G encoded in unary, and let p be an initial place and t be a transition.*

1. *The number of different simple expressions of length m is $B(m) \stackrel{def}{=} 2^{(|P|\cdot(c_{max}+2)\cdot m)+m}$.*
2. *It is possible to compute a symbolic representation of the set of markings coverable from some marking in the initial set $\mathbb{N}\cdot\{(p,0)\}$, as a finite set of simple expressions. I.e., one can compute simple expressions $S_1,\ldots,S_\ell$ s.t. $\bigcup_{1\le i\le \ell}[\![S_i]\!] = Cover(\mathbb{N}\cdot\{(p,0)\})$ and where $\ell \le 3\cdot B(2)$. Each of the S_i has length either 2 or 4.*
3. *Checking if there exists $M \in Cover(\mathbb{N}\cdot\{(p,0)\})$ with $M \longrightarrow_t$ can be done in $O(|P|\cdot c_{max})$ deterministic space.*

Proof. For Item 1 note that a simple expression is described by a word where some symbols have a Kleene star. There are $|\Sigma|^m$ different words of length m and 2^m possibilities to attach stars to symbols. Since the alphabet is $\Sigma \stackrel{\text{def}}{=} 2^{P\times[c_{max}+1]}$ and $|[c_{max}+1]| = c_{max}+2$, the result follows.

Towards Item 2, we can assume w.l.o.g. that our TPN is non-consuming by Lemma 2.7.1, and thus the region abstraction introduced in Section 2.7.3 applies. In particular, the initial set of markings $\mathbb{N}\cdot\{(p,0)\}$ is represented exactly by the expression $S_0 \stackrel{\text{def}}{=} \{(p,0)\}\emptyset^*$ where $\emptyset \in \Sigma$ is the symbol corresponding to the empty set. That is, we have $[\![S_0]\!] = \mathbb{N}\cdot\{(p,0)\}$ and thus $Cover([\![S_0]\!]) = Cover(\mathbb{N}\cdot\{(p,0)\})$.

The claimed expressions S_i are the result of iterating Algorithm 1 until a previously seen expression is revisited. Starting at $i = 0$ and $S_0 \stackrel{\text{def}}{=} \{(p,0)\}\emptyset^*$, each round will set S_{i+1}, S_{i+2} and S_{i+3} to the result of applying Algorithm 1 to S_i, and increment i to $i+3$.

Notice that then all S_i are simple expressions of length 2 or 4 and that in particular, all expressions with index divisible by 3 are of the form ab^* for $a, b \in \Sigma$. Therefore after at most $B(2)$ iterations, an expression S_ℓ is revisited (with $\ell \le 3B(2)$). Finally, an induction using Lemma 2.7.13 provides that $\bigcup_{1\le i\le \ell}[\![S_i]\!] = Cover(\mathbb{N}\cdot\{(p,0)\})$.

Towards Item 3, we modify the above algorithm for the ∃COVER problem with the sliding window technique. The algorithm is the same as above where instead of recording all the expressions $S_1, \ldots, S_\ell$, we only store the most recent ones and uses them to decide whether the transition t is enabled. If the index i reaches the maximal value of $3 \cdot B(2)$ we return unsuccessfully.

The bounded index counter uses $O(\log(B(2)))$ space; Algorithm 1 uses space $O(\log(B(5)))$ because it stores only simple expressions of length ≤ 5. The space required to store the three expressions resulting from each application of Algorithm 1 is $O(3 \cdot \log(B(4)))$. For every encountered simple expression we can check in logarithmic space whether the transition t is enabled by some marking in its denotation. Altogether the space used by our new algorithm is bounded by $O(\log(B(5)))$. By Item 1, this is $O(|P| \cdot (c_{max} + 2)) = O(|P| \cdot c_{max})$. □

Corollary 2.7.15. *The ∃COVER problem for TPN is* PSPACE*-complete.*

Proof. The PSPACE lower bound was shown in Theorem 2.6.3. The upper bound follows from Lemma 2.7.1 and Item 3 of Theorem 2.7.14. □

Chapter 3

Probabilistic Infinite-State Systems

In this chapter we define a mathematical formalism for representing probabilistic infinite state systems. We first provide a brief introduction to the core underlying probabilistic model on which all systems that we are going to study in this chapter are based, namely the Markov Decision Process. We briefly mention several general applications of MDPs in the real world, along with the most studied objectives that appear in the literature. The rest of this chapter is organized as follows. We define formally what Markov Chains and Markov Decision Processes are, how strategies are represented, and what classes of different strategies exist. We formally define the notion of objective, and mention the types of problems that are studied on MCs and MDPs, such as the *qualitative* and *quantitative* problems.We define objectives that we are going to use in our analysis, along with a brief survey of recent results on infinite state MDPs. In this chapter, we will not focus on the general literature for finite state MDPs since a lot of techniques that are used to solve problems on finite state systems fail on the infinite state case. In Section 3.2.1 we introduce the mathematical formalism for Vector Addition Systems with States (VASS), along with its probabilistic extension that we call VASS-MDPs. We show that a lot of qualitative problems such as sure/almost-sure/limit-sure problems are undecidable for VASS-MDPs. We then restrict our model to single sided VASS-MDPs. We present a result which has been published in [1], namely that the limit sure control state reachability problem for single sided VASS-MDPs where the controller can change the counter value is decidable.

In Section 3.4, we introduce the One-Counter Markov Decision Process, a model which was first studied in [28]. We mention related models, such as Solvency Games, Recursive Markov Decision Processes (RMDPs) or One-Counter Nets (OCNs). along with some known decidability problems. We then introduce the limit-sure selective

termination problem for OC-MDPs along with the almost sure $\{1,2,3\}$-parity problem (as studied in [61]) and show that almost sure $\{1,2,3\}$-parity problem for OC-MDPs is at least as hard as the limit-sure selective termination problem for OC-MDPs. Our main motivation for studying One-Counter Markov Decision Processes (OC-MDPs), along with its limit-sure selective termination problem comes from [28], where one would like to study its decidability, as it has been left open. We recall that neither the $\{1,2,3\}$-parity nor the selective termination problem for OC-MDPs that we will present here have particular applications. However, the $\{1,2,3\}$-parity problem is just the simplest subcase of the general parity problem, which is already 'very hard' in many ways. In other words, the $\{1,2,3\}$-parity problem is just a small subcase of general parity problem, but already very hard. Note that simpler subcases of parity exist in the Mostowski hierarchy, such as Büchi or co-Büchi objectives; see [61].

First, almost sure $\{1,2,3\}$-parity requires infinite memory on general MDPs (as shown in [61]). Second, even on OC-MDPs, decidability of almost sure $\{1,2,3\}$-parity is open, and (as shown in this thesis) it is at least as hard as the (also open) limit-sure selective termination problem.

3.1 Introduction

A *Markov decision process (MDP)* provides a mathematical representation of a system that has both randomized and controlled behaviour. In some situations a *controller* can choose among a certain set of actions to go to a certain successor state, whereas in other cases, this decision is based on a probability distribution among the set of possible actions that are currently available. This model has been extensively studied in the literature since it provides direct applicability to real world problems.

Markov decision processes (MDPs) [46] are a formal model for games on directed graphs, where certain decisions are taken by a strategic player (a.k.a. Player 1, or controller) while others are taken randomly (a.k.a. by nature, or the environment) according to pre-defined probability distributions. MDPs are thus a subclass of general 2-player stochastic games, and they are equivalent to 1.5-player games in the terminology of [32]. They are also called "games against nature".

A run of the MDP consists of a sequence of visited states and transitions on the graph. Properties of the system are expressed via properties of the induced runs. The most basic objectives are reachability (is a certain (set of) control-state(s) eventually visited?) and Büchi objectives (is a certain (set of) control-state(s) visited infinitely

often?).

Since a strategy of Player 1 induces a probability distribution of runs of the MDP, the objective of an MDP is defined in terms of this distribution, e.g., if the probability of satisfying a reachability/Büchi objective is at least a given constant. The special case where this constant is 1 is a key example of a qualitative objective. Here one asks whether Player 1 has a strategy that achieves an objective surely (all runs satisfy the property) or almost-surely (the probability of the runs satisfying the property is 1).

Most classical work on algorithms for MDPs and stochastic games has focused on finite-state systems (e.g., [46, 84, 34]), but more recently several classes of infinite-state systems have been considered as well. For instance, MDPs and stochastic games on infinite-state probabilistic recursive systems (i.e., probabilistic pushdown automata with unbounded stacks) [44] and on one-counter systems [28, 26] have been studied. Another infinite-state probabilistic model, which is incomparable to recursive systems, is a suitable probabilistic extension of Vector Addition Systems with States (VASS; a.k.a. Petri nets). While recursive systems use an unbounded pushdown stack, VASS have a finite number of unbounded counters holding natural numbers.

3.1.1 Preliminaries

The set of finite words over an alphabet Σ is denoted as Σ^*, and the set of infinite words are represented as Σ^ω. We define $\Sigma^+ \stackrel{\text{def}}{=} \Sigma^* \setminus \{\varepsilon\}$, where ε is the empty word. For a given word $w \in \Sigma$, we define $\mathit{length}(w)$ as the length of word w. If $w = \varepsilon$, then $\mathit{length}(w) = 0$. In the case where w is an infinite word, we have that $\mathit{length}(w) = +\infty$. Given a word $w \in \Sigma^*$, we denote by $w \downarrow n$ the prefix $w_0 w_1 w_2 \cdots w_{n-1}$ of w. For a word $w \in \Sigma^*$, we denote the individual letters of w by $w(0), w(1), \ldots$, where the indexing of the words starts at zero. Let $\mathbb{N}$ (resp. $\mathbb{Z}$) denote the set of nonnegative integers (resp. integers). For two integers i, j such that $i \leq j$ we use $[i..j]$ to represent the set $\{k \in \mathbb{Z} \mid i \leq k \leq j\}$. Given a set X and $n \in \mathbb{N} \setminus \{0\}$, X^n is the set of n-dimensional vectors with values in X. We use $\mathbf{0}$ to denote the vector such that $\mathbf{0}(i) = 0$ for all $i \in [1..n]$. The classical order on $\mathbb{Z}^n$ is noted $\leq$ and is defined by $\mathbf{v} \leq \mathbf{w}$ if and only if for all $i \in [1..n]$, we have $\mathbf{v}(i) \leq \mathbf{w}(i)$. We also define the operation $+$ over n-dimensional vectors of integers in the classical way (i.e., for $\mathbf{v}, \mathbf{v}' \in \mathbb{Z}^n$, $\mathbf{v} + \mathbf{v}'$ is defined by $(\mathbf{v} + \mathbf{v}')(i) = \mathbf{v}(i) + \mathbf{v}'(i)$ for all $i \in [1..n]$). Given a set S, we use S^* (respectively S^ω) to denote the set of finite (respectively infinite) sequences of elements of S. A probability distribution on a countable set X is a function $f : X \mapsto [0,1]$ such that $\sum_{x \in X} f(x) = 1$. We use $\mathcal{D}(X)$ to denote the set of all probability

distributions on X. Given $f \in \mathcal{D}(X)$, we let $Supp(f) = \{x \in X \mid f(x) > 0\}$ to be the support of f. We first recall some basic definitions from probability theory.

Definition 3.1.1. Let X be a set and define $\mathcal{X}$ to be a collection of subsets of X. Then, $\mathcal{X}$ is an *field* if

- $X \in \mathcal{X}$
- If $A \in \mathcal{X}$ then $\overline{A} \in \mathcal{X}$
- If $A \in \mathcal{X}$ and $B \in \mathcal{X}$ then $A \cup B \in \mathcal{X}$

Definition 3.1.2. An *experiment* is a procedure that can be infinitely repeated and has a well-defined (non-empty) set of possible outcomes, which is called a *sample space*. An experiment is called *random* if it has more than one possible outcome, and *deterministic* if it has a single possible outcome.

Definition 3.1.3 (σ-field)**.** A σ-*field* over a set Ω is a set $\mathcal{F} \subseteq 2^{\Omega}$ of subsets of Ω, where

- $\Omega \in \mathcal{F}$
- If $A \in \mathcal{F}$ then $\overline{A} \in \mathcal{F}$
- If $A_i \in F$ for every member of a countably indexed family $\{A_i : i \in I\}$, then $\bigcup_{i \in I} A_i \in \mathcal{F}$

Definition 3.1.4 (Probability measure)**.** Let Ω be a sample space and let $\mathcal{A}$ be a σ-field of subsets of Ω. We call the function $\mathbb{P} : \mathcal{A} \to [0,1]$ *probability measure* (a.k.a. *probability distribution*) on $(\Omega, \mathcal{A})$ if it satisfies the following conditions (a.k.a. *Kolmogorov axioms*):

1. $\mathbb{P}(\varnothing) = 0$ and $\mathbb{P}(\Omega) = 1$
2. $\mathbb{P}$ is σ-additive, i.e., for each sequence of disjoint sets $A_i \in \mathcal{A}$, $i = 1, 2, ...$, we have that
$$\mathbb{P}(\bigcup_{i=1}^{\infty} A_i) = \Sigma_{i=1}^{\infty} \mathbb{P}(A_i)$$

Definition 3.1.5 (Probability space)**.** A *probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ is a tuple made by a set of outcomes Ω (also known as *sample space*), a σ-algebra $\mathcal{F} \subseteq 2^{\Omega}$ of *events* over sample space Ω, and a *probability measure* $\mathbb{P} : \mathcal{F} \to [0,1]$.

Definition 3.1.6 (Markov Chain)**.** A Markov Chain (MC) $\mathcal{M} = (C, \to, p)$ is a tuple where C is a countable set of states, $\to \subseteq C \times C$ is a transition relation, and p is a function that assigns to every state $s \in C$ a positive probability distribution over the outgoing transitions of s.

A path is a finite or infinite sequence $\rho = c_1 c_2 \ldots$ of states such that $(c_i, c_{i+1}) \in \to$ holds for every index i. An infinite path is called a run. We write $w \in S^*$ to denote a finite path. We write $c \xrightarrow{x} c'$ to denote the fact that $(c, c') \in \to$ and $p(c)(c') = x$. We write $Runs_{\mathcal{M}}(w)$ to represent the set of infinite words wS^{ω}, namely the set of infinite paths with finite prefix $w \in S^*$. To every state s in S we assign the probability space $(Runs_{\mathcal{M}}(s), \mathcal{F}, \mathbb{P})$ of infinite paths which start at s, where $\mathcal{F}$ is the σ-field generated by all basic cones $Runs_{\mathcal{M}}(w)$, and $\mathbb{P} : \mathcal{F} \to [0,1]$ is the unique probability measure such that $\mathbb{P}(Runs_{\mathcal{M}}(w)) = \Pi_{i=1}^{length(w)-1} p_i$, where $w(i-1) \xrightarrow{p_i} w(i)$, for all $1 \leq i < length(w)$. From the Carathéodory's extension theorem [33], this defines a unique probability measure on all measurable subsets of runs.

Definition 3.1.7 (MDPs)**.** A *Markov Decision Process (MDP) M* is a tuple $\langle C, C_1, C_P, A, \to, p \rangle$ where: C is a countable set of states partitioned into C_1 and C_P (that is $C = C_1 \cup C_P$ and $C_1 \cap C_P = \emptyset$); A is a set of actions; $\to \subseteq C \times A \times C$ is a transition relation; $p : C_P \mapsto \mathcal{D}(C)$ is a partial function which assigns to some states in C_P probability distributions on C such that $p(c)(c') > 0$ if and only if $c \xrightarrow{a} c'$ for some $a \in A$.

Note that our definition is equivalent as seeing MDPs as games played between a non-deterministic player (Player 1) and a probabilistic player (Player P). The set C_1 contains the nondeterministic states (or states of Player 1) and the set C_P contains the probabilistic states (or states of Player P). Given two states c, c' in C, we write $c \to c'$ whenever there exists $a \in A$ such that $c \xrightarrow{a} c'$. We will say that a configuration $c \in C$ is a deadlock if there does not exist $c' \in C$ such that $c \to c'$. We use C_1^{df} (resp. C_P^{df}), to denote the states of Player 1 (resp. of Player P) which are not a *deadlock* (*df* stands here for deadlock free).

For a given state $c \in C$, we say that c' is a successor state of c whenever $c \to c'$. An MDP M is *finitely branching* if every state has only finitely many successors. Otherwise, it is called *infinitely branching*.

In our MDP framework, we interchangeably use the notion of paths with plays. Hence, for the purpose of presentation, in what follows the notion of infinite plays will correspond to runs.

A *play* of the MDP $M = \langle C, C_1, C_P, A, \rightarrow, p \rangle$ is either an infinite sequence of the form $c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ or a finite sequence $c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots \xrightarrow{a_{k-1}} c_k$. We will call the first kind of plays an *infinite play*, and the second one a *finite play*. A play is said maximal whenever it is infinite or it ends in a deadlock state. These latter plays are called deadlocked plays. We use Ω to denote the set of maximal plays. For a finite play $\rho = c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots \xrightarrow{a_{k-1}} c_k$, let $c_k = \mathit{last}(\rho)$. The notation Ω_1^{df} will denote the set of finite plays ρ such that $\mathit{last}(\rho) \in C_1^{df}$.

A *strategy* for Player 1 is a function σ that assigns each word $wv \in \Omega_1^{df}$ to a probability distribution over the set of outgoing transitions of v. Namely, if $\sigma(wv)(t) > 0$ then $t = (v,u) \in \rightarrow$, for some $u \in C$.

Intuitively, given a finite play ρ, which represents the history of the game so far, the strategy represents the choice of Player 1 among the different possible successor configurations from $\mathit{last}(\rho)$. We use Π to denote the set of all strategies for Player 1. Given a strategy $\sigma \in \Pi$, an infinite play $c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ *respects* σ if for every $k \in \mathbb{N}$, we have that if $c_k \in C_1$ then $c_{k+1} = \sigma(c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots c_k)$ and if $c_k \in C_P$ then $p(c_k)(c_{k+1}) > 0$. We define finite plays that respect σ similarly. Let $\mathtt{Plays}(M, c, \sigma) \subseteq \Omega$ be the set of all maximal plays of M that start from c and that respect σ.

Note that once a starting state $c_0 \in C$ and a strategy σ have been chosen, the MDP M is reduced to an ordinary stochastic process, i.e., a Markov chain, that we call $M(\sigma)$, whose set of states are C^* and $xu \xrightarrow{p_i} xut$ if and only if $u \rightarrow t$ and one of the following conditions hold, either

- $u \in C_P$ and $p(u)(t) = p_i$ or
- $u \in C_1$ and $\sigma(xu)$ assigns p_i to the transition $u \rightarrow t$

We define an *objective* (also known as *event*) $\mathcal{A} \subseteq \Omega$ as a measurable set of plays and we use $\mathbb{P}(M, c, \sigma, \mathcal{A})$ to denote the probability of objective $\mathcal{A}$ starting from $c \in C$ under strategy σ. The notation $\mathit{Val}(M, c, \mathcal{A})$ will be used to represent the maximal probability of event $\mathcal{A}$ starting from c which is defined as follows $\mathit{Val}(M, c, \mathcal{A}) = \sup_{\sigma \in \Pi} \mathbb{P}(M, c, \sigma, \mathcal{A})$. We will say that a strategy σ for player 1 is *optimal* from the starting state c for the event $\mathcal{A}$ if $\mathit{Val}(M, c, \mathcal{A}) = \mathbb{P}(M, c, \sigma, \mathcal{A})$. We will say that a strategy σ for player 1 is ε-*optimal* from the starting state c the objective $\mathcal{A}$ if $\mathbb{P}(M, c, \sigma, \mathcal{A}) \geq 1 - \varepsilon$.

3.1.2 Types of strategies

We say that strategy $\sigma : \Omega_1^{df} \mapsto C$ is *memoryless* if for every $\rho, \rho' \in \Omega_1^{df}$ and $x \in C_1$, $\sigma(\rho x) = \sigma(\rho' x)$. A strategy is *deterministic* if given $\rho \in \Omega_1^{df}$ and $x \in C_1$, $\sigma(\rho x)$ assigns probability 1 to some transition. A strategy which is memoryless and deterministic will be abbreviated as *MD*. Strategies that are not (necessarily) memoryless are called history-dependent (a.k.a. *H*). Moreover, strategies which are not necessarily deterministic are called randomized (a.k.a. *R*). Note that a deterministic strategy is a special case of a randomized strategy. At every step, a player can choose a unique move with probability 1. A strategy is history-randomized (a.k.a. *HR*) if it is both *H* and *R*. Let us denote Π^{MD} and Π^{HR} by the set of of all possible *MD* and *HR* strategies, respectively. It is easy to observe that the set of all possible strategies for the controller (Player 1) (which is denoted as Π) is the same as the set of history-randomized strategies (denoted as Π^{HR}).

For an MDP $M = \langle C, C_1, C_P, A, \rightarrow, p \rangle$, we can construct a strategy by using a probabilistic transducer $\mathcal{T} = \langle \mathtt{Mem}, m_0, \pi_u, \pi_s \rangle$, where $\mathtt{Mem}$ is a countable set (representing the memory of the strategy), $m_0 \in \mathtt{Mem}$ is called the initial memory mode and C (the set of states of the MDP) is the input and output alphabet. The memory mode of the transducer is updated by a probabilistic transition function $\pi_u : \mathtt{Mem} \times C \rightarrow \mathcal{D}(\mathtt{Mem})$. The probabilistic successor function $\pi_s : \mathtt{Mem} \times C_1 \rightarrow \mathcal{D}(C)$ outputs the next successor, where $c' \in \mathit{Supp}(\pi_s(m,c))$ implies that $c \rightarrow c'$. We lift the functions $\pi_u : \mathcal{D}(\mathtt{Mem}) \times C \rightarrow \mathcal{D}(\mathtt{Mem})$ and $\pi_s : \mathcal{D}(\mathtt{Mem}) \times C_1 \rightarrow \mathcal{D}(\mathtt{Mem})$, in the natural way. We extend π_u to paths by $\pi_u(m, \varepsilon) = m$ and $\pi_u(m, c_0 \rightarrow c_1 \rightarrow \ldots c_n) = \pi_u(\pi_u(c_0 \ldots c_{n-1}, m), c_n)$. The strategy $\sigma_{\mathcal{T}}$ that is determined by the transducer $\mathcal{T}$ is defined as $\sigma_{\mathcal{T}}(c_0 \rightarrow c_1 \rightarrow \ldots c_n) \stackrel{\text{def}}{=} \pi_s(c_n, \pi_u(m, c_0 \rightarrow c_1 \rightarrow \ldots c_{n-1}, m_0))$. Note that a history dependent (H) strategy σ has finite memory if there exists a transducer $\mathcal{T}$ with memory $\mathtt{Mem}$ such that $\sigma = \sigma_{\mathcal{T}}$ and $|\mathtt{Mem}| < \infty$. Otherwise, the strategy needs infinite memory. Moreover, a memoryless strategy can be implemented by a probabilistic transducer where $|M| = 1$.

3.1.3 Types of analysis for problems on MCs and MDPs

Computational problems can be classified into *qualitative* and *quantitative* problems. Informally, in a *qualitative* framework, we would like to decide problems whether a certain objective (a.k.a. event) holds with probability one, or whether the complement of this objective holds with probability zero. In *quantitative* problems, we are interested in computing the optimal probability that the desired objective holds. For instance,

we want to compute if a certain objective holds with probability p, and decide exact questions, such as $p \leq \alpha$, where α is a constant. Another version of the quantitative problem involves computing the maximal probability of up-to arbitrary precision. This is called an approximation of the quantitative problem.

They are presented as in the following.

1. *Qualitative* analysis. Let $\mathcal{M}$ be a MDP (or a MC) $\mathcal{M}$, with objective *Obj*, and a set of strategies Π [1]. We want to decide whether from a starting state c_0 there exists a strategy $\sigma \in \Pi$ such that the objective *Obj* holds almost surely, i.e., decide whether

$$\exists \sigma \in \Pi . \mathbb{P}(\mathcal{M}, c_0, \sigma, Obj) = 1$$

This (decision) problem is known as the qualitative almost-sure decision problem for the objective E, with respect to strategy σ. Clearly, we can consider the complementary problem, when one would like to decide whether

$$\exists \sigma \in \Pi . \mathbb{P}(\mathcal{M}, c_0, \sigma, \overline{Obj}) = 0$$

where we define the objective $\overline{Obj} \stackrel{\text{def}}{=} \Omega \setminus Obj$. In the case where σ exists, one would want to synthesise it, namely to construct it explicitly. Hence, in this case, one would not refer to a decision problem, but to a problem of constructing a strategy, in the case where this exists. Another qualitative decision problem (very relevant to this thesis project) is the *qualitative* **limit-sure** decision problem for an objective *Obj*, i.e decide whether

$$\sup_{\sigma \in \Pi} \mathbb{P}(\mathcal{M}, c_0, \sigma, Obj) = 1$$

This is equivalent with the following formulation, i.e decide whether

$$\forall \varepsilon > 0 . \exists \sigma_\varepsilon \in \Pi . \mathbb{P}(\mathcal{M}, c_0, \sigma_\varepsilon, Obj) \geq 1 - \varepsilon$$

It is worth mentioning that for finite state Markov Decision Processes, the limit sure and almost sure decision problems coincide [28]. In the general case, where the state space of the MDP (and MC) is infinite, this may not be the case, depending on the objective. In particular, one can see in Section 3.4.7, that for an infinite state MDP finitely represented as a OC-MDP, for the objectives such as selective termination [28] limit sure case does not imply almost sure selective termination, whereas the other direction is valid.

[1] Note that we do not specify the nature of strategies here, i.e memoryless, history randomized, etc.

There exist other qualitative decision problems beyond the scope of this project, such as the *qualitative witness positivity* decision problem ([]). For an objective *Obj*, one would like to decide whether there exists a strategy $\sigma \in \Pi$ such that $\mathbb{P}(\mathcal{M}, c_0, \sigma, \overline{Obj}) > 0$.

2. *Quantitative* analysis.

   - *Exact quantitative* decision problems. Given an MDP, an initial control state c_0, a rational $\alpha \in \mathbb{Q}$, one would like to decide whether there exists a strategy $\sigma \in \Pi$, such that $\mathbb{P}(\mathcal{M}, c_0, \sigma, Obj) \geq \alpha$. We can further apply this framework for the expected value of a random variable X, i.e., where one would like to answer questions such as whether there exists a strategy $\sigma \in \Pi$, such that $\mathbb{E}(\mathcal{M}, c_0, \sigma, (X)) \geq \alpha$. In both of these cases, we are talking about a maximization problem, i.e., one would like to make the probability of achieving a certain objective (or the expectation of a random variable X) as high as possible. Consider the converse problem, where one would like to ask questions such as whether there exists a strategy $\sigma \in \Pi$, such that $\mathbb{P}(\mathcal{M}, c_0, \sigma, Obj) \leq \alpha$ (or the case of $\mathbb{E}(\mathcal{M}, c_0, \sigma, X) \leq \alpha$). In these cases, we are talking about a minimization problem, where one would like to make the probability of achieving a certain objective *Obj* (or the expectation of a random variable X), as low as possible. In practice, it happens that these problems are computationally hard, hence, we will use *approximation* problems.

   - *Quantitative* ε*-approximation* analysis. Given an MDP $\mathcal{M}$, an initial starting state c_0, an objective *Obj*, let us define $v^* \stackrel{\text{def}}{=} \sup_{\sigma \in \Pi} \mathbb{P}(\mathcal{M}, c_0, \sigma, Obj)$. Given a rational $\varepsilon > 0$, one would like to compute an ε-approximate value $v \in \mathbb{Q}$, such that $|v^* - v| < \varepsilon$. Intuitively, the two quantities v^* and v would like to be as close as possible, up to ε-error of precision. Conversely, one would like to define $v^* \stackrel{\text{def}}{=} \inf_{\sigma \in \Pi} \mathbb{E}(\mathcal{M}, c_0, \sigma, X)$ and compute a rational value v such that the previous inequality holds.

     Furthermore, one would like to find an ε-optimal strategy σ such that $|v^* - \mathbb{P}(\mathcal{M}, c_0, \sigma, Obj)| < \varepsilon$. Conversely, given a random variable X, one would like to find an ε-optimal strategy σ such that $|v^* - \mathbb{E}(\mathcal{M}, c_0, \sigma, X)| < \varepsilon$.

3.1.4 Orderings

In what follows, we will define upward and downward-closed sets with respect to an ordering, and state a result by American algebraist Leonard Eugene Dickson (i.e., Dickson lemma [35]) that we will later use in Section 3.3. Note that preorders (quasi-orderings) and well-quasi-orderings have been defined in Chapter 2, as in Definition 2.1.1.

Definition 3.1.8 (Upward/Downward-closed sets)**.** A set of states F is *upward-closed* with respect to ordering $\preceq$ if and only if

$$\langle q_1, v_1 \rangle \in F \wedge \langle q_1, v_1 \rangle \preceq \langle q_2, v_2 \rangle \implies \langle q_2, v_2 \rangle \in F$$

Conversely F is *downward-closed* with respect to $\preceq$ if and only if

$$\langle q_2, v_2 \rangle \in F \wedge \langle q_1, v_1 \rangle \preceq \langle q_2, v_2 \rangle \implies \langle q_1, v_1 \rangle \in F$$

Note that the complement of a downward-closed set is upward-closed and vice-versa.

Lemma 3.1.1 (Dickson's lemma)**.** [35] For every infinite sequence $X_1 X_2 X_3 \ldots$ of vectors of $\mathbb{N}^k$ there exists an infinite sequence $i_1 < i_2 < i_3 < \ldots$ of indices such that $X_{i_1} \leq X_{i_2} \leq X_{i_3} \leq \ldots$.

3.1.5 Objectives on Countable MDPs

We will define the objectives that are most frequently studied on countable state MDPs, such as reachability, safety and parity, using the Linear Temporal Logic (LTL) framework. We then define a class of finite state MDPs that are equipped with *rewards* (a.k.a. costs) on each transition, a model which is equivalent to an infinite state MDP which encodes the state and the accumulated reward.

Linear Temporal Logic (LTL) is a modal temporal logic where formulas are built from a finite set of propositions, using the Boolean logical connectives $\neg$, $\wedge$, $\vee$, along with *temporal connectives*, in order to reason about conditions that may hold in the future. There exists two types of temporal connectives, such as the unary `Next` operator (represented as $\bigcirc$) and the binary `Until` operator (represented as `U`). For example, if γ is an LTL formula, then the formula **True** `U` γ means 'γ holds eventually', being represented as $\Diamond\gamma$. Also, the formula $\neg(\Diamond\neg\gamma)$ denotes that 'γ always holds', being represented as $\Box\gamma$. A fully formal mathematical representation of how formulas are defined in this model can be found in [16].

Given an MDP $M = \langle C, C_1, C_P, A, \to, p \rangle$, formulas are interpreted on the transition system $\langle C, \to \rangle$. Let $[\![\phi]\!]_c \subseteq cC^\omega$, representing the set of plays which start in c and satisfy formula ϕ. It has been shown that this set of plays is measurable [87] and write $\mathbb{P}(\mathcal{M}, c_0, \sigma, \phi)$ to represent $\mathbb{P}(\mathcal{M}, c_0, \sigma, [\![\phi]\!]_c)$. We define $[\![\phi]\!] \stackrel{\text{def}}{=} \bigcup_{c \in C} [\![\phi]\!]_c$.

We now define the reachability and safety objective.

Definition 3.1.9 (Reachability)**.** Given an MDP $M = \langle C, C_1, C_P, A, \to, p \rangle$ and a set $T \subseteq C$ of target states, we say that a play $\rho \stackrel{\text{def}}{=} c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ satisfies the *reachability* condition if and only if there exists an index $i \in \mathbb{N}$ such that $c_i \in T$. Let $[\![\Diamond T]\!]$ denote the set of plays which satisfy the reachability condition.

Definition 3.1.10 (Safety)**.** Given an MDP $M = \langle C, C_1, C_P, A, \to, p \rangle$ and a set $T \subseteq C$ of target states, we say that $\rho \stackrel{\text{def}}{=} c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ satisfies the *reachability* condition if and only if for every index $i \in \mathbb{N}$, $c_i \neq T$. Let $[\![\Box \neg T]\!]$ denote the set of plays which satisfy the safety condition.

In a parity objective, every state has a *priority*, out of a finite set of priorities that are natural numbers. An infinite play satisfies the parity objective if and only if the maximal priority that is visited infinitely often is *even*.

Definition 3.1.11 (Parity objective)**.** Given a countable state MDP $M = \langle C, C_1, C_P, A, \to, p \rangle$, let $Col \subseteq \mathbb{N}$ be a finite set of *colors*. We define a *priority* (a.k.a. color) function $\lambda : C \to Col$, mapping each state to a natural number. For $n \in \mathbb{N}$, we define the set $\oplus \stackrel{\text{def}}{=} \{=, \leq, \geq, \geq\}$ and given $S \subseteq C$, let $[S]^{\lambda \oplus n} \stackrel{\text{def}}{=} \{s \in S \mid \lambda(s) \oplus n\}$ in order to represent the set of states in S which have priority $\oplus n$. We define the parity objective as

$$\mathtt{PAR}(\lambda) \stackrel{\text{def}}{=} [\![\bigvee_{i \in Col} (\Box \Diamond [S]^{\lambda = 2i} \wedge \Diamond \Box [S]^{\lambda \leq 2i})]\!]$$

In other words, $\mathtt{PAR}(\lambda)$ consists of the set of plays such that the maximal priority that occurs infinitely often along the play is even.

We can classify different parity objectives by restricting the codomain of the priority function λ. An exhaustive list of subclasses of parity problems can be encountered in the Mostowski hierarchy [71]. We denote Col-PAR for restricted parity objectives where $Col \subseteq \mathbb{N}$. Here, we present some of the most frequently used variants of parity objectives, such as Büchi and co-Büchi objectives. We can represent a Büchi objective as a $\{1,2\}$-PAR objective, whereas a co-Büchi objective is represented as a $\{0,1\}$-PAR objective.

Remark 3.1.2. For an MDP $M = \langle C, C_1, C_P, A, \to, p \rangle$ and a set $T \subseteq C$ of target states, the $\{1,2\}$-PAR and $\{0,1\}$-PAR objectives subsume the reachability objective by defining the priority function $\lambda(c) = 1 \iff c \notin T$, where $c \in C$. Also, both $\{1,2\}$-PAR and $\{0,1\}$-PAR objectives subsume the safety objective by defining the priority function $\lambda(c) = 1 \iff c \in T$, where $c \in C$.

Remark 3.1.3. [48] For finite state MDPs, *MD* strategies are sufficient for every type of qualitative and quantitative parity objectives. In other words, if one can satisfy the parity objective (using a qualitative or quantitative analysis), then strategies that achieve this ignore the history of play and they are deterministic.

Since the underlying structures of the models that we will study in Section 3.2 and Section 3.4, namely VASS-MDPs and OC-MDPs, are based on countably (infinite) state MDPs, we would like to briefly present some state of the art results that have been published in recent literature.

3.1.6 Countable state MDPs

The work of [61] studies *general countable state MDPs* with *parity objectives*, and special cases when the number of colors are bounded in the Mostowski hierarchy. In finite state MDPs with parity objectives, there always exist an optimal memoryless deterministic (MD) strategy. However, when the MDP is infinite, in general, this is not the case, i.e., optimal strategies may not exist. The most important result is the fact that even for the case of *finitely branching* countably infinite state MDPs, the strategies which satisfy almost-sure $\{1,2,3\}$-parity requires infinite memory (Theorem 3.1.4).

Theorem 3.1.4 (cf. Theorem 1 of [61])**.** *There exists a finitely branching MDP $\mathcal{M}$ with color function λ, initial state c_0 such that*

1. *for every strategy $\sigma \in \Pi^{HR}$, we have $\mathbb{P}(\mathcal{M}, c_0, \sigma, \{1,2,3\}\text{-PAR}(\lambda)) = 0$*
2. *there exists a strategy $\sigma \in \Pi^{HD}$ such that $\mathbb{P}(\mathcal{M}, c_0, \sigma, \{1,2,3\}\text{-PAR}(\lambda)) = 1$*

Therefore, optimal (and even almost-surely) winning) and ε-optimal strategies require infinite memory for $\{1,2,3\}$-PAR, even in finitely branching MDPs.

In the MDP $\mathcal{M}$ from Section 3.1.6, all states of the form c_i have $\lambda(c_i) = 1$, for every $i \in \mathbb{N}$ and are controlled. All states under the form r_i have $\lambda(r_i) = 2$, for every $i \in \mathbb{N}$, and are stochastic, whereas state t is controlled and has $\lambda(t) = 3$. The probabilities are labelled on the corresponding transitions.

Intuitively, for any history-randomized (*HR*) strategy there exists a strictly positive probability of visiting state t with priority 3 between consecutive visits to c_0. In the long run, unless for the case where only states c_i are visited (i.e., the player does not use any transition towards r_i), state t will be visited almost-surely. This implies that the $\{1,2,3\}$-PAR objective will be satisfied with probability 0. However, one can build a strategy to almost surely satisfy the $\{1,2,3\}$-PAR objective in the following way. At the k-th visit to state c_0, one can use the path $c_0c_1\dots c_k$ and then switch to state r_k, where $s_k \to r_k$. By going along this path one can make the probability of visiting t smaller and smaller, between the previous and succesive visits to c_0. Therefore, in the long run, the probability of visiting state t is 0 and hence the largest color that is visited infinitely often is 2, satisfying the $\{1,2,3\}$-PAR objective. However, one can not store infinitely many ks to perform this strategy, thus needing infinite memory. A detailed representation can be found in the proof of Theorem 3.1.4, from [61].

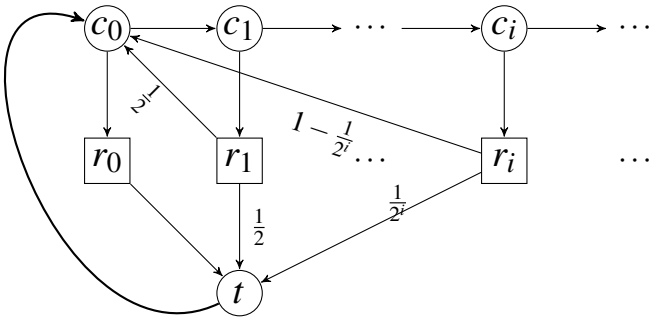


Figure 3.1: A finitely branching MDP $\mathcal{M}$ where starting from state c_0 one can satisfy the $\{1,2,3\}$ objective *almost surely*, as presented in [61]. The controlled (non-deterministic) states are drawn as circles, whereas the probabilistic states are drawn as squares. For every $i \in \mathbb{N}$, every (controlled) state c_i has priority 1, whereas every (probabilistic) state r_i has priority 2. The state t is controlled, having priority 3.

It is shown that even for infinitely branching MDPs under Büchi objective, optimal strategies, if they exist, can be chosen *MD* (Theorem 12, [61]). Moreover, for finitely branching MDPs under $\{0,1,2\}$-PAR objective, if there exist optimal strategies, they can be chosen *MD* (Theorem 16, [61]). Furthermore, it is shown that for co-Büchi objectives, ε-optimal strategies can be chosen MD (Theorem 19, [61]).

We now present recent literature about several combined objectives on *finite state*

MDPs with rewards.

3.1.7 Finite state MDPs with Rewards

A finite state MDP with rewards consists of a finite MDP where each transition is labelled by an integer. Let $\mathcal{M} = \langle C, C_1, C_P, A, \rightarrow, p \rangle$ be a finite MDP with reward function $r : \rightarrow \rightarrow \mathbb{Z}$. Note that the first arrow $\rightarrow$ in the signature of r denotes its domain, whereas $\mathbb{Z}$ represents its codomain.

An *energy condition* on MDP $\mathcal{M}$ is defined with respect to reward function r in the following way. Given an initial energy level $k \in \mathbb{N}$, an infinite play $\rho = c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ satisfies the k-energy condition if and only if for every finite prefix, $k + \Sigma_{i=0} r(c_i, c_{i+1}) \geq 0$. Let $\mathtt{EN}(k)$ be the set of all infinite plays that satisfy the k- energy objective.

We define *Mean Payoff* conditions with respect to cost function $r : \rightarrow \rightarrow \mathbb{Z}$. An infinite play $\rho = c_0 \xrightarrow{a_1} c_1 \xrightarrow{a_2} c_2 \xrightarrow{a_3} \ldots$ satisfies the positive mean-payoff condition if and only if $\liminf_{n \to \infty} \Sigma_{i=0}^{n-1} \frac{r(c_i, c_{i+1})}{n} > 0$. Let $\mathtt{MP}_{>0}$ be the set of infinite plays that satisfy the positive mean-payoff condition.

We recall some results about combined objectives, namely the energy-parity ($\mathtt{EN}(k) \cap \mathtt{PAR}$) and mean-payoff parity ($\mathtt{MP}_{>0} \cap \mathtt{PAR}$).

In energy-parity objectives, one would like to take into account the remaining stored energy of the system (such as a battery), combined with a parity condition. It has been shown both in [68] and that the almost-sure *energy-parity* objective for finite state MDPs with rewards is decidable and is in $NP \cap coNP$, being solved in pseudo-polynomial time. Moreover, in order to achieve almost sure energy parity objective, one would need to use infinite-memory. Moreover, it is shown [68] that the *limit-sure energy-parity* problem does *not* coincide with the almost-sure one. However, the problem is still in $NP \cap coNP$. Other variants of the energy objectives exists as well such as the k-storage objective, where the energy level must not drop by a certain amount fixed by the controller [68]. The *almost-sure mean-payoff-parity* problem is decidable in polynomial time [31], hence PTIME-complete.

3.1.8 Infinite state MCs

We would like to present some recent results about infinite state MCs, in particular a subclass called *decisive* MCs. Intuitively, a MC is decisive with respect to a given set of target (final) states T if it almost surely eventually reaches either T or a state from

which T can no longer be reached. Note that by construction, all finite MCs are *trivially* decisive. Moreover, for *some* particular classes of infinite state MCs, this is also the case.

- By (Lemma 3.4, [7]), all infinite state MCs which have a finite attractor are decisive w.r.t T. An attractor represents a set of states from which one can reach them almost surely from every state of the MC.
- By (Lemma 3.5, [7]), all infinite state MCs which are *globally coarse* are decisive w.r.t. T. A MC is *globally coarse* w.r.t. T if there exists $\theta > 0$ such that from every state, the probability of reaching T is either 0 or $\geq \theta$.

3.2 VASS-MDPs

In this section we study the decidability of limit-sure reachability for infinite-state MDPs that are induced by suitable probabilistic extensions of Vector Addition Systems with States that we call VASS-MDPs.

Most quantitative objectives in probabilistic VASS are either undecidable, or the solution is at least not effectively expressible in $(\mathbb{R}, +, *, \leq)$ [7]. It is easy to show that, for general VASS-MDPs, even the simplest of these problems, (almost) sure reachability, is undecidable (see Section 3.2.3). In particular, we focus on *single-sided* VASS-MDPs, which are split into two monotone subclasses: 1-VASS-MDPs and P-VASS-MDPs. In 1-VASS-MDPs, only Player 1 can modify counter values while the probabilistic player can only change control-states, whereas for P-VASS-MDPs it is vice-versa. Still these two models induce infinite-state MDPs. Unlike for finite-state MDPs, it is possible that the value of the MDP, in the game theoretic sense, is 1, even though there is no single strategy that achieves value 1. For example, there can exist a family of strategies σ_ε for every $\varepsilon > 0$, where playing σ_ε ensures a probability $\geq 1 - \varepsilon$ of reaching a given target state, but no strategy ensures probability 1. In this case, one says that the reachability property holds limit-surely, but not almost-surely (i.e., unlike in finite-state MDPs, almost-surely and limit-surely do not coincide in infinite-state MDPs).

For our decidability result of the limit-sure reachability problem in 1-VASS-MDP, we use an algorithm which at each iteration reduces the dimension of the considered VASS while preserving the limit-sure reachability properties.

This work has been published in [1]. Furthermore, in [1] it has also been shown that even for P-VASS-MDPs, all sure/almost-sure/limit-sure reachability/Büchi problems

are still undecidable. However, in the deadlock-free subclass of P-VASS-MDPs, the sure reachability/Büchi problems become decidable (while the other problems remain undecidable). In contrast, for 1-VASS-MDPs, the sure/almost-sure reachability problem and the sure/almost-sure Büchi problem are decidable, by reducing them to the model-checking problem over VASS of a restricted fragment of the modal μ-calculus that has been proved to be decidable in [9].

3.2.1 Qualitative Analysis of VASS-Induced MDPs

Probabilistic Vector Addition Systems with States have been studied, e.g., in [7]. Here we extend this model with non-deterministic choices by a controller. We call this new model VASS-MDPs. We first recall the definition of Vector Addition Systems with States.

Definition 3.2.1 (Vector Addition System with States)**.** For $n > 0$, an n-dimensional Vector Addition System with States (VASS) is a tuple $S = \langle Q, T \rangle$ where Q is a finite set of control states and $T \subseteq Q \times \mathbb{Z}^n \times Q$ is the transition relation labelled with vectors of integers.

In the sequel, we will not always make precise the dimension of the considered VASS. Configurations of a VASS are pairs $\langle q, \mathbf{v} \rangle \in Q \times \mathbb{N}^n$. Given a configuration $\langle q, \mathbf{v} \rangle$ and a transition $t = \langle q, \mathbf{z}, q' \rangle$ in T, we will say that t is *enabled* at $\langle q, \mathbf{v} \rangle$, if $\mathbf{v} + \mathbf{z} \geq \mathbf{0}$. Let then $\mathtt{En}(q, \mathbf{v})$ be the set $\{t \in T \mid t \text{ is enabled at } \langle q, \mathbf{v} \rangle\}$. In case the transition $t = \langle q, \mathbf{z}, q' \rangle$ is enabled at $\langle q, \mathbf{v} \rangle$, we define $t(q, \mathbf{v}) = \langle q', \mathbf{v}' \rangle$ where $\mathbf{v}' = \mathbf{v} + \mathbf{z}$. An n-dimensional VASS S induces a labelled transition system $\langle C, T, \rightarrow \rangle$ where $C = Q \times \mathbb{N}^n$ is the set of configurations and the transition relation $\rightarrow \subseteq C \times T \times C$ is defined as follows: $\langle q, \mathbf{v} \rangle \xrightarrow{t} \langle q', \mathbf{v}' \rangle$ iff $\langle q', \mathbf{v}' \rangle = t(q, \mathbf{v})$. VASS are sometimes seen as programs manipulating integer variables, a.k.a. counters. When a transition of a VASS changes the i-th value of a vector $\mathbf{v}$, we will sometimes say that it modifies the value of the i-th counter. We now show in which manner we add probability distributions to VASS, and obtain a VASS-MDP. Conceptually, the finite state space of a VASS-MDP is partitioned into controlled and probabilistic states, and every transition is assigned a positive natural number, denoted as *transition weight*. Note that we do not explicitly introduce probability distributions on the outgoing transitions at this level, but we do it at the level of the equivalent infinite state MDP via encoding VASS-MDP configurations. In particular, for every probabilistic configuration c, one can define the probability of going from c to configuration c' by taking the summation of weights of the corresponding

transitions in the VASS-MDP, divided by the total weights of the enabled transitions at that particular configuration c. One can easily check that in this way the probability function for the infinite state MDP is well-defined.

Definition 3.2.2 (VASS-MDP)**.** A VASS-MDP is a tuple $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ where $\langle Q, T \rangle$ is a VASS for which the set of control states Q in partitioned into Q_1 and Q_P and $\tau : T \mapsto \mathbb{N} \setminus \{0\}$ is a partial function assigning to each transition a weight which is a positive natural number.

We use $T_1 \subseteq T$ (respectively $T_P \subseteq T$) to denote the subsets of transitions leaving from a nondeterministic state (respectively a probabilistic state). Hence $T = T_1 \cup T_P$ with $T_1 \subseteq Q_1 \times \mathbb{Z}^n \times Q$ and $T_P \subseteq Q_P \times \mathbb{Z}^n \times Q$. A VASS-MDP $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ induces an MDP $M_S = \langle C, C_1, C_P, T, \to, p \rangle$ where: $\langle C, T, \to \rangle$ is the labelled transition system associated to the VASS $\langle Q, T \rangle$; $C_1 = Q_1 \times \mathbb{N}^n$ and $C_P = Q_P \times \mathbb{N}^n$; and for all $c \in C_P$ and $c' \in C$ if $c \to c'$, the probability of going from c to c' is defined as follows:

$$p(c)(c') = \frac{\sum_{\{t \mid t(c)=c'\}} \tau(t)}{\sum_{t \in \mathtt{En}(c)} \tau(t)} \tag{3.1}$$

and we have $p(c)(c') = 0$ in case $c \not\to c'$. Note that the MDP M_S is well-defined: when defining $p(c)(c')$ in the case $c \to c'$, there exists at least one transition in $\mathtt{En}(c)$ and consequently the sum $\sum_{t \in \mathtt{En}(c)} \tau(t)$ is never equal to 0. Also, we could have restricted the weights to be assigned only to transitions leaving from a control state in Q_P since we do not take into account the weights assigned to the other transitions. A deadlock free VASS-MDP is a VASS-MDP whose underlying VASS is deadlock free.

Finally, as in [79] or [9], we will see that to gain decidability it is useful to restrict the power of the nondeterministic player or of the probabilistic player by restricting their ability to modify the counters values and hence letting them only choose a control location. This leads to the two following definitions: a *P-VASS-MDP* is a VASS-MDP $\langle Q, Q_1, Q_P, T, \tau \rangle$ such that for all $\langle q, \mathbf{z}, q' \rangle \in T_1$, we have $\mathbf{z} = \mathbf{0}$ and a *1-VASS-MDP* is a VASS-MDP $\langle Q, Q_1, Q_P, T, \tau \rangle$ such that for all $\langle q, \mathbf{z}, q' \rangle \in T_P$, we have $\mathbf{z} = \mathbf{0}$. In other words, in a P-VASS-MDP, Player 1 cannot change the counter values when taking a transition and in a 1-VASS-MDP, it is Player P which cannot perform such an action.

3.2.2 Verification Problems for VASS-MDPs

We consider qualitative verification problems for VASS-MDPs, taking as objectives control-state reachability and repeated reachability. To simplify the presentation, we

consider a single target control state $q_F \in Q$. However, our positive decidability results easily carry over to sets of target control states (while negative ones trivially do). Note, however, that asking to reach a *fixed target configuration* like $\langle q_F, \mathbf{0} \rangle$ is a very different problem [7]. Let $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ be a VASS-MDP and M_S its associated MDP. Given a control state $q_F \in Q$, we denote by $[\![\Diamond q_F]\!]$ the set of infinite plays $c_0 \cdot c_1 \cdots$ and deadlocked plays $c_0 \cdots c_l$ of M_S for which there exists an index $k \in \mathbb{N}$ such that $c_k = \langle q_F, \mathbf{v} \rangle$ for some $\mathbf{v} \in \mathbb{N}^n$. Similarly, $[\![\Box\Diamond q_F]\!]$ characterizes the set of infinite plays $c_0 \cdot c_1 \cdots$ of M_S for which the set $\{i \in \mathbb{N} \mid c_i = \langle q_F, \mathbf{v} \rangle \text{ for some } \mathbf{v} \in \mathbb{N}^n\}$ is infinite. Since M_S is an MDP with a countable number of configurations, we know that the sets of plays $[\![\Diamond q_F]\!]$ and $[\![\Box\Diamond q_F]\!]$ are measurable (for more details see for instance [16]), they are hence events for M_S. Given an initial configuration $c_0 \in Q \times \mathbb{N}^n$ and a control state $q_F \in Q$, we consider the following questions for the VASS-MDP S:

1. The *sure reachability problem*: Does there exist a strategy $\sigma \in \Sigma$ such that $\texttt{Plays}(M_S, c_0, \sigma) \subseteq [\![\Diamond q_F]\!]$?
2. The *almost-sure reachability problem*: Does there exist a strategy $\sigma \in \Sigma$ such that $\mathbb{P}(M_S, c_0, \sigma, [\![\Diamond q_F]\!]) = 1$?
3. The *limit-sure reachability problem*: Does $Val(M_S, c_0, [\![\Diamond q_F]\!]) = 1$?
4. The *sure repeated reachability problem*: Does there exist a strategy $\sigma \in \Sigma$ such that $\texttt{Plays}(M_S, c_0, \sigma) \subseteq [\![\Box\Diamond q_F]\!]$?
5. The *almost-sure repeated reachability problem*: Does there exist a strategy $\sigma \in \Sigma$ such that $\mathbb{P}(M_S, c_0, \sigma, [\![\Box\Diamond q_F]\!]) = 1$?
6. The *limit-sure repeated reachability problem*: Does $Val(M_S, c_0, [\![\Box\Diamond q_F]\!]) = 1$?

Note that sure reachability implies almost-sure reachability, which itself implies limit-sure reachability, but not vice-versa, as shown by the counterexamples in Figure 3.2. The same holds for repeated reachability. Furthermore for the sure problems, probabilities are not taken into account, and thus these problems can be interpreted as the answer to a two player reachability game played on the transition system of S. Such games have been studied for instance in [79, 2, 9]. Finally, VASS-MDPs subsume deadlock-free VASS-MDPs and thus decidability (resp. undecidability) results carry over to the smaller (resp. larger) class.

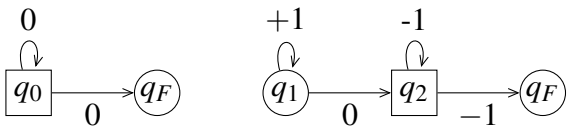


Figure 3.2: Two 1-dimensional VASS-MDPs. The circles (resp. squares) are the control states of Player 1 (resp. Player P). All transitions have the same weight 1. From $\langle q_0, 0\rangle$ the state q_F is reached almost-surely, but not surely, due to the possible run with an infinite loop at q_0 (which has probability 0). From $\langle q_1, 0\rangle$, the state q_F can be reached limit-surely, by a family of strategies that repeats the loop at q_1 more and more often, but not almost-surely (or surely), since every strategy has a chance of getting stuck at state q_2 with counter value zero.

3.2.3 Undecidability in the General Case

It was shown in [2] that the sure reachability problem is undecidable for two player VASS. From this we can deduce that the sure reachability problem is undecidable for VASS-MDPs. We now present a similar proof to show the undecidability of the almost-sure reachability problem for VASS-MDPs.

For all of our undecidability results we use reductions from the undecidable control-state reachability problem for Minsky machines. A Minsky machine is a tuple $\langle Q, T\rangle$ where Q is a finite set of states and T is a finite set of transitions manipulating two counters, say x_1 and x_2. Each transition is a triple of the form $\langle q, x_i = 0?, q'\rangle$ (counter x_i is tested for 0) or $\langle q, x_i := x_i + 1, q'\rangle$ (counter x_i is incremented) or $\langle q, x_i := x_i - 1, q'\rangle$ (counter x_i is decremented) where $q, q' \in Q$. Configurations of a Minsky machine are triples in $Q \times \mathbb{N} \times \mathbb{N}$. The transition relation $\Rightarrow$ between configurations of the Minsky machine is then defined in the obvious way. Given an initial state q_I and a final state q_F, the control-state reachability problem consists in asking whether there exists a sequence of configurations $\langle q_I, 0, 0\rangle \Rightarrow \langle q_1, v_1, v_1'\rangle \Rightarrow \ldots \Rightarrow \langle q_k, v_k, v_k'\rangle$ with $q_k = q_F$. This problem is known to be undecidable [70]. W.l.o.g. we assume that Minsky machines are deadlock-free and deterministic (i.e., each configuration has always a unique successor) and that the only transition leaving q_F is of the form $\langle q_F, x_1 := x_1 + 1, q_F\rangle$.

We now show how to reduce the control-state reachability problem to the almost-sure and limit-sure reachability problems in deadlock-free VASS-MDPs. From a Minsky machine, we construct a deadlock-free 2-dim VASS-MDP for which the control states

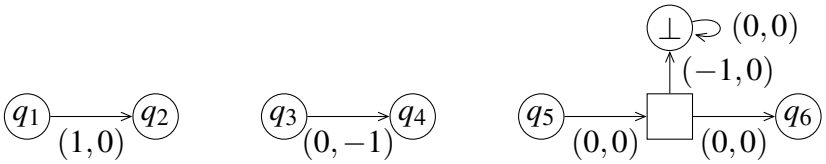


Figure 3.3: Encoding $\langle q_1, x_1 := x_1 + 1, q_2 \rangle$ and $\langle q_3, x_2 := x_2 - 1, q_4 \rangle$ and $\langle q_5, x_1 = 0?, q_6 \rangle$

of Player 1 are exactly the control states of the Minsky machine. The encoding is presented in Figure 3.3 where the circles (resp. squares) are the control states of Player 1 (resp. Player P), and for each edge the corresponding weight is 1. The state $\perp$ is an absorbing state from which the unique outgoing transition is a self loop that does not affect the values of the counters. This encoding allows us to deduce our first result.

Theorem 3.2.1. *The sure, almost-sure and limit-sure (repeated) reachability problems are undecidable problems for deadlock-free VASS-MDPs.*

In the special case of 1-dimensional VASS-MDPs, the sure and almost-sure reachability problems are decidable [28].

3.2.4 Probabilistic Vector Addition Systems with States (PVASS)

A *probabilistic VASS (PVASS)*[7] is a VASS where every state is probabilistic. One can immediately notice the difference with regard to the VASS-MDP model. There, some control states were non-deterministic, whereas here, it is not the case.

From a PVASS, one can derive an infinite state MC in a similar manner as in Section 3.2. We recall some of the decidability questions for PVASS. Given PVASS $\mathcal{V}$ and its associated infinite state MC $\mathcal{M}_{\mathcal{V}}$, in [7] it has been shown the following

1. $\mathcal{M}_{\mathcal{V}}$ is decisive.
2. The *approximate quantitative reachability* problem is decidable when the set of target states T is upward closed.
3. The *qualitative reachability* problem is undecidable if the set of target states T is a general upward closed set (i.e represented by its finitely many minimal elements)
4. The *qualitative repeated reachability* problem (for probability 1) is decidable if the set of targets T is upward closed. For qualitative-probability 0 repeated reachability, the problem is open.

3.3 Limit-Sure Control State Reachability for 1-VASS-MDP

We consider a slightly more general version of the limit-sure reachability problem with a set $X \subseteq Q$ of target states instead of a single state q_F, i.e., the standard case corresponds to $X = \{q_F\}$.

Limit sure control state reachability for 1-VASS-MDPs

Input: a 1-VASS-MDP $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ of dimension $n \geq 0$, initial configuration $c_0 = \langle q_0, v \rangle \in Q \times \mathbb{N}^n$, a set of target states $X \subseteq Q$

Question: Is $Val(M_S, c_0, [\![\Diamond X]\!]) = 1$?

Definition 3.3.1. We extend the set of natural numbers $\mathbb{N}$ to $\mathbb{N}_* = \mathbb{N} \bigcup \{*\}$ by adding an element $* \notin \mathbb{N}$ with $* + j = * - j = *$ and consider the set of vectors $\mathbb{N}_*^d$. The projection of a vector z in $\mathbb{N}^d$ by eliminating components that are indexed by a natural number k is defined by

$$proj_k(z)(i) = \begin{cases} z(i) & \text{if } i \neq k \\ * & \text{if } otherwise \end{cases}$$

Let Q_c represent control-states which are indexed by a color. The coloring functions $col_i : Q \to Q_c$ create colored copies of control-states by $col_i(q) = q_i$.

Given a 1-VASS-MDP $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ of dimensions d, an index $k \leq d$ and a color i, the projection is defined as:

$$Proj_k(M, d, i) = (col_i(Q), col_i(Q_1), col_i(Q_P), proj_{k,i}(T), \tau)$$

where $proj_{k,i}(T) = \{proj_{k,i}(t) | t \in T\}$ is the projection of the set of transitions T and $proj_{k,i}(t) = (col_i(x), proj_k(op), col_i(y))$ is the projection of transition t by removing component k and coloring the states x and y with color i.

We define the functions $state : Q \times \mathbb{N}^d \to Q$ and $count : Q \times \mathbb{N}^d \to \mathbb{N}^d$ s.t for a configuration $c_i = (q, \mathbf{v})$, where $q \in Q$ and $\mathbf{v} \in \mathbb{N}^d$ we have that $state(q, \mathbf{v}) = q$ and $count(q, \mathbf{v}) = \mathbf{v}$. For any 2 configurations c_1 and c_2, we write $c_1 \prec c_2$ to denote that $state(c_1) = state(c_2)$, and there exists a nonempty set of indexes I where for every $i \in I$, $count(c_1)(i) < count(c_2)(i)$, whereas for every index $j \notin I$, $0 < j \leq d$, $count(c_1)(j) = count(c_2)(j)$.

Algorithm 2 reduces the dimension of the limit-sure reachability problem for 1-VASS-MDP, by a construction resembling the Karp-Miller tree [60]. It takes as input

a 1-VASS-MDP S of some dimension $d > 0$ with a set of target states X. It outputs a new 1-VASS-MDP S' of dimension $d-1$ and a new set of target states X' such that M_S can limit-surely reach X iff $M_{S'}$ can limit-surely reach X'. In particular, in the base case where $d-1=0$, the new system S' has dimension zero and thus induces a finite-state MDP $M_{S'}$, for which limit-sure reachability of X' coincides with almost-sure reachability of X', that is known to be decidable in polynomial time. Algorithm 2 starts by exploring all branches of the computation tree of S (and adding them to S' as the so-called initial *uncolored part*) until it encounters a configuration that is either (1) equal to, or (2) strictly larger than a configuration encountered previously on the same branch. In case (1) it just adds a back loop to the point where the configuration was encountered previously. In case (2), it adds a modified copy of S (identified by a unique color) to S'. This so-called colored subsystem is similar to S except that those counters than have strictly increased along the branch are removed. The intuition is that these counters could be pumped to arbitrarily high values and thus present no obstacle to reaching the target. Since the initial uncolored part is necessarily finite (by Dickson's Lemma) and each of the finitely many colored subsystems only has dimension $d-1$ (since a counter is removed; possibly a different one in different colored subsystems), the resulting 1-VASS-MDP S' has dimension $d-1$. The set of target states X' is defined as the union of all appearances of states in X in the uncolored part, plus all colored copies of states from X in the colored subsystems.

Lemma 3.3.1. Algorithm 2 terminates.

Proof. Algorithm 2 explores an unfolding of the computation tree of S, which is finitely branching since $|T|$ is finite. The number of counters is fixed, and therefore, by Dickson's Lemma, $(\mathbb{N}^d, \preceq)$ is a well quasi ordering. Therefore, on every branch we eventually satisfy either the condition of line 20 or of line 8. In the former case, a loop in the derived system S' is created, and the exploration of the current branch stops. In the latter case, a finitary description of a new colored (possibly infinite-state) subsystem is added to S' by adding finitely many states, transitions and configurations to Q', T' and X', respectively. Also in this case, the exploration of the current branch stops. Since the exploration is finitely branching, and every branch eventually stops, the algorithm terminates. □

We now illustrate a run of Algorithm 2 using the following example.

Algorithm 2 Reducing the dimension of the limit-sure reachability problem.

Input: $S = (Q, Q_1, Q_P, T, \tau)$ 1-VASS-MDP, dimension $d > 0$, $c_0 = (q_0, v) \in Q \times \mathbb{N}^d$
$X \subseteq Q$ - set of target states

Output: $S' = (Q', Q'_1, Q'_P, T', \tau')$; $c'_0 = (q'_0, 0)$; $X' \subseteq Q'$; $\lambda : Q' \to ((Q \bigcup Q_c) \times \mathbb{N}^d_*)$

$Q' \leftarrow \varnothing$; $Q'_1 \leftarrow \varnothing$; $Q'_P \leftarrow \varnothing$; $T' \leftarrow \varnothing$; $\tau' \leftarrow \varnothing$
new(q'); $q'_0 \leftarrow q'$; $\lambda(q') \leftarrow c_0$; $Q' \leftarrow \{q'\}$; $i \leftarrow 0$
if $state(\lambda(q')) \in Q_1$ **then** $Q'_1 \leftarrow \{q'\}$ **else** $Q'_P \leftarrow \{q'\}$
ToExplore $\leftarrow \{q'\}$
while ToExplore $\neq \varnothing$ **do**
  Pick and remove a $q \in$ ToExplore
  if $\exists q'$. q' is previously on the same brach as q and $\lambda(q') \prec \lambda(q)$ **then**
    get indexes I in which the counter is increasing
    pick and remove the first index k from I
    $i \leftarrow i + 1$; // increase color index
    new(q");
    $\lambda(q'') \leftarrow (col_i(state(\lambda(q))), proj_k(count(\lambda(q))))$
    if $state(\lambda(q)) \in Q_1$ **then** $Q'_1 \leftarrow Q'_1 \bigcup \{q''\}$ **else** $Q'_P \leftarrow Q'_P \bigcup \{q''\}$
    $T' \leftarrow T' \bigcup \{(q, 0, q'')\}$; $\tau'(\langle q, 0, q' \rangle) = 1$
    $Q'_1 \leftarrow Q'_1 \bigcup col_i(Q_1)$; $Q'_P \leftarrow Q'_P \bigcup col_i(Q_P)$; $T' \leftarrow T' \bigcup proj_{k,i}(T)$;
    $X' \leftarrow X' \bigcup col_i(X)$; $\tau' \leftarrow \tau' \cup \tau_{k,i}$
  else
    for every $t = \langle x, z, y \rangle \in T$ such that $t \in \mathbf{En}(\lambda(q))$ **do**
      if $\exists q'$. q' is previously on the same branch as q and $t(\lambda(q)) = \lambda(q')$ **then**
        $T' \leftarrow T' \bigcup \{(q, z, q')\}$
      else
        new(q'); $\lambda(q') \leftarrow t(\lambda(q))$
        $T' \leftarrow T' \bigcup \{(q, z, q')\}$; $\tau'(\langle q, z, q' \rangle) \leftarrow \tau(t)$
        if $state(\lambda(q')) \in Q_1$ **then** $Q'_1 \leftarrow Q'_1 \bigcup \{q'\}$ **else** $Q'_P \leftarrow Q'_P \bigcup \{q'\}$
        if $state(\lambda(q')) \in X$ **then** $X' \leftarrow X' \bigcup \{q'\}$
        $ToExplore \leftarrow ToExplore \bigcup \{q'\}$
      end if
    end for
  end if
end while

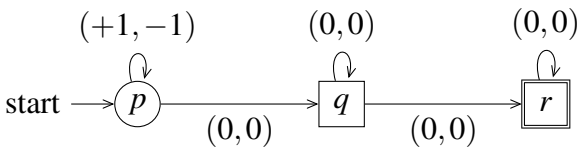


Figure 3.4: A 1-VASS-MDP S . The circled state are controlled, whereas the squared ones are probabilistic. All transitions have weight 1.

Example 3.3.1. Let $S = \langle Q, Q_1, Q_P, T, \tau \rangle$ be a 1-VASS-MDP of dimension $d = 2$, where $Q \stackrel{\text{def}}{=} \{p,q,r\}$, $Q_1 \stackrel{\text{def}}{=} \{x\}$, $Q_P \stackrel{\text{def}}{=} \{y,z\}$, $c_0 = \langle x,(2,3)\rangle$, and $X \stackrel{\text{def}}{=} \{r\}$, and set of transitions T as in Figure 3.4. All transitions have weight 1.

- Lines 1-5: $q_0' \leftarrow a$, $\lambda(a) \leftarrow \langle p,(2,3)\rangle$ $Q_1' \leftarrow \{a\}$, $\mathit{ToExplore} \leftarrow \{a\}$
- Line 6: $\mathit{ToExplore} \leftarrow \varnothing$
- Lines 18-26: let $t = \langle p,(+1,0),p\rangle \in T$; create b;

  $\lambda(b) \leftarrow \langle p,(3,2)\rangle$; $T' \leftarrow \{a,(+1,-1),b\}$; $Q_1' \leftarrow \{a,b\}$; $\mathit{ToExplore} \leftarrow \{b\}$;

  Let $t = \langle p,(0,0),q\rangle \in T$; create c; $\lambda(c) \leftarrow \langle q,(2,3)\rangle$; $Q_P' \leftarrow \{c\}$; $T' \leftarrow T' \cup \{a,(0,0),c\}$; $\mathit{ToExplore} \leftarrow \{c,b\}$;
- Line 6: remove c; $\mathit{ToExplore} \leftarrow \{b\}$
- Line 18-26: let $t = \langle q,(0,0),r\rangle$; create d;

  $\lambda(d) \stackrel{\text{def}}{=} \langle r,(2,3)\rangle$; $T' \leftarrow T' \cup \{c,(0,0),d\}$; $X' \leftarrow \{d\}$; $\mathit{ToExplore} \stackrel{\text{def}}{=} \{d,b\}$
- Lines 6 and 18-20 : remove d; $\mathit{ToExplore} \leftarrow \{b\}$; let $t = \langle r,(0,0),r\rangle$; $T' \leftarrow T' \cup \{d,(0,0),d\}$
- Line 6-16: remove b; $\mathit{ToExplore} \leftarrow \varnothing$; $i \leftarrow 1$; create p_1; $\lambda(p_1) = \langle p_1,(*,2)\rangle$ $Q_1' \leftarrow Q_1' \cup \{p_1\}$; $Q_P' \leftarrow \{q_1,r_1\}$; $X' \leftarrow \{r_1,d\}$
- Halt.

Remark 3.3.2. Limit sure control state reachability for 1-VASS-MDPs is EXPSPACE-hard, since it is at least as hard as control state reachability for VASS (which is has been shown to be EXPSPACE-hard in [39]).

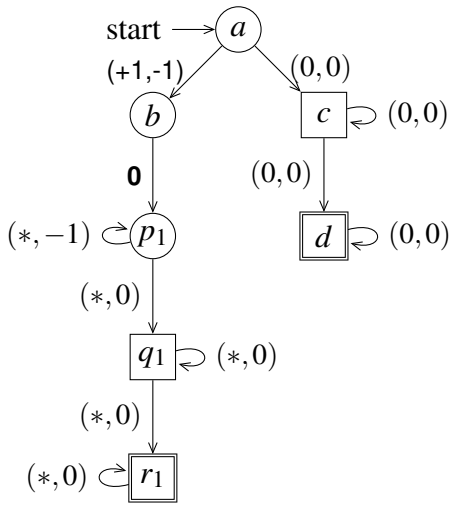


Figure 3.5: Given as input 1-VASS-MDP S from Example 3.3.1, Algorithm 2 produces the 1-VASS-MDP S' described as above. The circled control states are controlled, whereas the squared ones are probabilistic. The double edge control states are targets.

The following two lemmas show the correctness of Algorithm 2. Let $S = (Q, Q_1, Q_P, T, \tau)$ be 1-VASS-MDP of dimension $d > 0$ with initial configuration $c_0 = (q_0, v)$ and $X \subseteq Q$ a set of target states. Let $S' = (Q', Q'_1, Q'_P, T', \tau')$ with initial configuration $c'_0 = (q'_0, 0)$ and set of target states $X' \subseteq Q'$ be the $(d-1)$ dimensional 1-VASS-MDP produced by Algorithm 2.

Lemma 3.3.3. $Val(M_S, c_0, [\![\Diamond X]\!]) = 1 \implies Val(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1.$

Proof. Let us assume that $Val(M_S, c_0, [\![\Diamond X]\!]) = 1$. Therefore, there exists a family of strategies that make the probability of reaching X arbitrarily close to 1. In other words, $\forall \varepsilon, \exists \sigma_\varepsilon, \mathbb{P}(M_S, c_0, \sigma_\varepsilon, [\![\Diamond X]\!]) \geq 1 - \varepsilon$. For every $\varepsilon > 0$ we use the strategy σ_ε of player 1 on M_S to construct a copycat strategy σ'_ε for the game on $M_{S'}$ that starts in $c'_0 = (q_0, \mathbf{0})$, such that it achieves $[\![\Diamond X']\!]$ with probability $\geq 1 - \varepsilon$.

The strategy σ'_ε will use the same moves on $M_{S'}$ as σ_ε on M_S, which is possible due to the way how $M_{S'}$ is constructed from M_S by Algorithm 2. By construction, for every reachable configuration in M_S there is a corresponding configuration in $M_{S'}$, and this correspondence can be maintained stepwise in the moves of the game.

For the initial uncolored part of $M_{S'}$, this is immediate, since S' is derived from the unfolding of the game tree of S. The correspondence is expressed by the function λ. Each current state of $M_{S'}$ is labeled by the corresponding current configuration of M_S.

In the colored subsystems, the corresponding configuration in system $M_{S'}$ is a projection of a configuration in M_S. For any transition $t \in T$ that is controlled by

player 1 from a configuration in M_S, there exists a transition $t' \in T'$ that belongs to player 1 in the corresponding configuration in $M_{S'}$, such that this transition leads to the corresponding state. This is achieved by the projection and the fact that the 1-VASS-MDP game is monotone w.r.t. player 1, i.e., larger configurations always benefit the player (by allowing the same moves or even additional moves).

We now show a property on how probabilistic transitions in M_S and $M_{S'}$ correspond to each other: For every probabilistic transition $t \in T$ from a configuration in M_S, there exists a probabilistic transition $t' \in T'$ in the corresponding configuration in $M_{S'}$, and vice-versa, such that these transitions have the same probability. In particular, a configuration in $M_{S'}$ does not allow any additional probabilistic transitions compared to its corresponding configuration in M_S (though it may allow additional transitions controlled by player 1).

The first part of this statement follows from the monotonicity of the projection function and the monotonicity of the transitions w.r.t. the size of the configurations. For the second part we need to show that for every probabilistic transition $t' = (col_i(x), proj_k(op), col_i(y)) \in T'$ from a configuration in $M_{S'}$, there exists a probabilistic transition $t = (x, op, y) \in T$ in the corresponding configuration in S, such that the probabilities of these transitions are equal. This latter fact holds only because we are considering 1-VASS-MDP, where only the player can change the counters, whereas the probabilistic transitions can only change the control-states. I.e., the 'larger' projected configurations in $M_{S'}$ do not enable additional probabilistic transitions, since in 1-VASS-MDP these only depend on the control-state.

Therefore, by playing in $M_{S'}$ using strategy σ'_ε with the same moves as σ_ε plays in M_S, we reach the same corresponding configurations in $M_{S'}$ with the same probability values as in M_S. Since the definition of the target set X' in S' includes all configurations corresponding to configurations in X on S, it follows from $\mathbb{P}(M_S, c_0, \sigma_\varepsilon, [\![\Diamond X]\!])) \geq 1-\varepsilon$ that $\mathbb{P}(M_{S'}, c'_0, \sigma'_\varepsilon, [\![\Diamond X']\!]) \geq 1-\varepsilon$. Since, by assumption above, this holds for every $\varepsilon > 0$, we obtain $Val(M_{S'}, c'_0, [\![\Diamond X']\!])) = 1$. □

Lemma 3.3.4. $Val(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1 \implies Val(M_S, c_0, [\![\Diamond X]\!]) = 1$.

Proof. We use the assumed family of strategies on $M_{S'}$ that witnesses the property $Val(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1$ to synthesize a family of strategies on M_S that witnesses $Val(M_S, c_0, [\![\Diamond X]\!]) = 1$.

First we establish some basic properties of the system S'. It is a 1-VASS-MDP of dimension $d-1$ with initial configuration c'_0, and consists of several parts. The

initial uncolored part induces a finite-state MDP. Moreover, S' contains finitely many subsystems of distinct colors, where each subsystem is a 1-VASS-MDP of dimension $d-1$ obtained from S by projecting out one component of the integer vector. For color i, let $k(i)$ be the projected component of the vector (see line 10 of the algorithm). Each colored subsystem of dimension $d-1$ induces an MDP that may be infinite-state (unless $d=1$, in which case it is finite-state).

Note that colored subsystems are not reachable from each other, i.e., a color, once reached, is preserved. Each colored subsystem has its own initial configuration (created in lines 12-13 of Alg. 2). Let m be the number of colors in S' and r_i the initial configuration of the subsystem of color i (where $0 \leq i \leq m-1$).

Let's now consider only those colored subsystems in which the target set X' can be reached limit-surely, i.e., let $J = \{i : 0 \leq i \leq m-1 \mid \mathbb{P}^+(M_{S'}, r_i, [\![\Diamond X']\!]) = 1\}$ be the set of good colors and let $R = \{r_j \mid j \in J\}$, and $\bar{R} = \{r_j \mid j \notin J\}$.

Further, let X'_f be the restriction of X' to the finite uncolored part of S' (i.e., only those parts added in line 26 of Alg. 2).

We now establish the existence of certain strategies in subsystems of S'. These will later serve as building blocks for our strategies on M_S.

Since we assumed that $Val(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1$, there exists a family of strategies that makes the probability of reaching X' arbitrarily close to one. In particular, they must also make the probability of reaching configurations in $\bar{R}$ arbitrarily close to zero. Thus we obtain $Val(M_{S'}, c'_0, [\![\Diamond X'_f \cup R]\!]) = 1$, i.e., we can limit-surely reach $X'_f \cup R$. Since, for this objective, only the finite uncolored part of $M_{S'}$ is relevant, this is a problem for a finite-state MDP and limit-surely and almost-surely coincide. So there exists a partial strategy σ, for the uncolored part of $M_{S'}$, such that, starting in c'_0, we almost-surely reach $X'_f \cup R$, i.e., $\mathbb{P}(M_{S'}, c'_0, \sigma, [\![\Diamond X'_f \cup R]\!]) = 1$.

In each of the good colored subsystems we can limit-surely reach X', i.e., for every $r_j \in R$ we have $Val(M_{S'}, r_i, [\![\Diamond X']\!]) = 1$. So for every $\varepsilon > 0$ there exists a strategy σ_i^ε such that $\mathbb{P}(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!]) \geq 1-\varepsilon$. Consider the computation tree of the game on $M_{S'}$ from r_i when playing according to σ_i^ε and its restriction to some finite depth d. Let $\mathbb{P}_d(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!])$ be the probability that the objective is reached already during the first d steps of the game. We have $\mathbb{P}_d(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!]) \leq \mathbb{P}(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!])$, but $\lim_{d\to\infty} \mathbb{P}_d(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!]) = \mathbb{P}(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!]) \geq 1-\varepsilon$. Thus for every color $i \in J$ and every $\varepsilon > 0$ there exists a number $d(i,\varepsilon)$ s.t. $\mathbb{P}_{d(i,\varepsilon)}(M_{S'}, r_i, \sigma_i^\varepsilon, [\![\Diamond X']\!]) \geq 1-2\varepsilon$.

Since configurations of $M_{S'}$ are obtained by projecting configurations of M_S, we can go the reverse direction by replacing the missing component in an $M_{S'}$ configuration

by a given number. Given an $M_{S'}$-configuration r_i and a number $d(i,\varepsilon)$ we obtain an M_S-configuration $s_i(d(i,\varepsilon))$ by replacing the missing $k(i)$-th component of r_i by $d(i,\varepsilon)$. Let $\alpha \in \mathbb{N}$ be the maximal constant appearing in any transition in S, i.e., the maximal possible change in any counter in a single step. Since a single step in M_S can only change a counter by $\leq \alpha$, the $k(i)$-th component of $s_i(\alpha * d(i,\varepsilon))$ cannot be exhausted during the first $d(i,\varepsilon)$ steps of the game on M_S starting at $s_i(\alpha * d(i,\varepsilon))$. Thus we can use the same strategy σ_i^ε in the game from $s_i(\alpha * d(i,\varepsilon))$ on M_S and obtain $\mathbb{P}(M_S, s_i(\alpha * d(i,\varepsilon)), \sigma_i^\varepsilon, [\![\Diamond X]\!]) \geq 1 - 2\varepsilon$. Intuitively, the number $\alpha * d(i,\varepsilon)$ is big enough to allow playing the game for sufficiently many steps to make the probability of success close to 1.

Using the strategy σ above and the strategies σ_i^ε, we now define a new family of strategies σ_ε for every $\varepsilon > 0$ for the game on M_S from c_0. Given $\varepsilon > 0$, we let $d(\varepsilon) = \alpha * \max_{i \in J} d(i,\varepsilon)$ (a number that is big enough for each projected component).

Playing from c_0 in M_S, the strategy σ_ε behaves as follows. First it plays like strategy σ in the corresponding game from c_0' on $M_{S'}$. (Function λ connects the corresponding configurations in the two games.) When the game in $M_{S'}$ reaches a configuration r_i then there are two cases: If the configuration in M_S is $\geq s_i(d(\varepsilon))$ then σ_ε henceforth plays like σ_i^ε, which ensures to reach the target X with probability $\geq 1 - 2\varepsilon$. Otherwise, the configuration in M_S is still too small to switch to σ_i^ε. In this case, σ_ε continues to play like σ plays from the previously visited smaller configuration in the uncolored part of $M_{S'}$ (see line 8 of the algorithm). This is possible, because the game is monotone and larger configurations always benefit Player 1. So the game on M_S continues with a configuration that is larger (at least on component $k(i)$) than the corresponding game on $M_{S'}$, i.e., component $k(i)$ is pumped. Since we know that σ on $M_{S'}$ will almost surely visit X_f' or R, we obtain that σ_ε on M_S will almost surely eventually visit X or some configuration $\geq s_i(d(\varepsilon))$ for $i \in J$ (and from there achieve to reach the target with probability $\geq 1 - 2\varepsilon$). Since every weighted average of probabilities $\geq 1 - 2\varepsilon$ is still $\geq 1 - 2\varepsilon$, we obtain $\mathbb{P}(M_S, c_0, \sigma_\varepsilon, [\![\Diamond X]\!]) \geq 1 - 2\varepsilon$ and thus $Val(M_S, c_0, [\![\Diamond X]\!]) = 1$. □

Theorem 3.3.5. *The limit-sure reachability problem for 1-VASS-MDP is decidable.*

Proof. Let $S = (Q, Q_1, Q_P, T, \tau)$ be 1-VASS-MDP of dimension $d > 0$ with initial configuration $c_0 = (q_0, v)$ and $X \subseteq Q$ a set of target states. We show decidability of $Val(M_S, c_0, [\![\Diamond X]\!]) = 1$ by induction on d. *Base case* $d = 0$. If S has 0 counters then M_S is a finite-state MDP and thus limit sure reachability coincides with almost sure reachability, which is decidable.

Inductive step. We apply Algorithm 2, which terminates by Lemma 3.3.1, and obtain a new instance of the 1-VASS-MDP limit sure reachability problem of dimension $d-1$: $S' = (Q', Q'_1, Q'_P, T', \tau')$ with initial configuration $c'_0 = (q'_0, 0)$ and set of target states $X' \subseteq Q'$. By Lemma 3.3.3 and Lemma 3.3.4, we have $\mathbb{P}^+(M_S, c_0, [\![\Diamond X]\!]) = 1 \Leftrightarrow \mathbb{P}^+(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1$. By induction hypothesis, $\mathbb{P}^+(M_{S'}, c'_0, [\![\Diamond X']\!]) = 1$ is decidable and the result follows. □

Below, in Figure 3.6, we summarize the results regarding all (control state) reachability problems related to VASS-MDPs, as presented in [1]. The ✗ mark represents the fact that the corresponding problem is undecidable, whereas the ✓ mark states that it is decidable. The ‘df‘ word is an abbreviation for ‘deadlock-free‘.

Reachability	P-VASS-MDP	df P-VASS-MDP	1-VASS-MDP
Sure	✗	✓	✓
Almost-sure	✗	✗	✓
Limit-sure	✗	✗	✓
Sure repeated	✗	✓	✓
Almost-sure repeated	✗	✗	✓
Limit-sure repeated	✗	✗	open

Figure 3.6: Control state reachability results for VASS-MDPs

3.4 One-Counter Markov Decision Processes (OC-MDPs)

One-counter Markov Decision Processes (OC-MDPs) are probabilistic variants of one-counter automata (OCA), which in turn are extensions of finite state automata with an unbounded counter. Equivalently, OC-MDPs can be viewed as extensions of Quasi-Birth-Death Processes (QBD) with a controller [14].

Several studies has been made in order to solve computational problems on classes of counter machines for which reachability is decidable, such as one counter automata [70], Petri nets and Vector Addition Systems with States (VASS) [67], [64], [77]. Initially, a counter could be incremented or decremented by one unit or remain the same. However, different problems has been studied in recent years, involving adding a certain constant to a counter [22] or adding parameters that are integers [25].

One counter processes (OCPs) operate on pushdown automata whose alphabet contains only one symbol. The reachability problem for pushdown automata can be solved in polynomial time [23]. In recent years, there has been multiple novel results related to verification of OCPs. For example, it has been shown in [51] that both model checking over OCPs with the temporal logic **EF** where formulas are represented as directed acyclic graphs (DAGs) and problems such as weak bisimilarity checking against finite systems are P^{NP}-complete. Their result is based on the membership problem under a fragment of Presburger Arithmetic, which it is shown that is P^{NP}-complete. We recall that P^{NP} represents a class of all problems that can be solved on a deterministic polynomial time Turing machine with access to an oracle from NP. Moreover, [51] shows that there exists a fixed **EF** formula (i.e., a finite system) where verification over OCPs is hard for $P^{NP[log]}$, where $P^{NP[log]}$ represents the class of all problems that can

be solved on a deterministic polynomial Turing machine which is allowed to perform $O(log(n))$ many queries to an oracle from NP. In the case where the system is fixed, the complexity drops to P.

One-Counter Markov decision processes (OC-MDPs) are a class of infinite state MDPs that are generated by finite-state automata that possess a single unbounded counter. Informally, an OC-MDP is a finite directed graph whose vertices are called *control states* and edges specify *transitions* between control states. A control state may be non-deterministic (i.e., *controlled* by Player 1) or probabilistic, where there exists a probability distribution over the set of outgoing edges. Every edge in the directed graph can increase/decrease the current counter value by one unit or leave it unchanged.. We denote *configurations* as pairs under the form $\langle p, i \rangle$, where p is a control state of the directed graph and i is the current counter value.

Every OC-MDP $\mathcal{V}$ induces two different types of countably infinite state (finitely branching) MDPs where the state space is made of configurations that encode control states of $\mathcal{V}$ and counter values - *with boundary* and *boundaryless*. An infinite state MDP *with boundary* encodes control states of $\mathcal{V}$ and counters that are natural numbers only, hence not allowing negative values. This is the model on which we are going to focus in detail in this section. Conversely, the *boundaryless* infinite state MDP encodes control states of $\mathcal{V}$ and counter values that are integers. Different objectives can be specified on these types, giving birth to different computational analysis.

The goal of the controller is to maximize the probability (respectively, optimize the expected value of an objective function) on the set of plays on the induced infinite state MDP. For example, one would like to study objectives such as reaching a configuration *for the first time* with counter value zero. This is known as a *termination* problem. Note that we do not make any constraint on the control states that are visited. A *selective termination* problem is a termination problem applied on a particular set of control states, which we call *targets*. In other words, given a set of control states T of a OC-MDP, the selective termination objective requires to reach a configuration with counter value 0 in a control state of T. The limit-sure selective termination problem requires to achieve the selective termination objective with probability arbitrarily close to 1. Our main motivation for studying limit-sure selective termination problem comes from [28], where one would like to study its decidability, as it has been left open. We would like to establish a connection between limit-sure selective termination for OC-MDPs and another (hard) problem. A suitable candidate for this is the almost sure $\{1,2,3\}$-parity problem.

In the OC-MDP framework - in order to consider the parity problem - all control states of the system are colored, i.e., they are labelled by a natural number. The almost-sure $\{1,2,3\}$-parity problem asks whether with probability 1 the maximal color that is visited infinitely often is 2. We recall that the $\{1,2,3\}$-parity problem is just a subcase of general parity problem, but already very hard. As presented in [61], simpler subcases of parity exist in the Mostowski hierarchy, such as Büchi or co-Büchi objectives. Note that as stated in [61], almost sure $\{1,2,3\}$-parity requires infinite memory on general MDPs. It remains open whether the almost sure $\{1,2,3\}$-parity and the limit-sure selective termination problems are decidable.

Our contribution is based on two results:

1. For the limit-sure selective termination objective, memoryless deterministic (MD) strategies are *sufficient*. In other words, for a OC-MDP $\mathcal{V}$, if one can achieve limit-sure selective termination on $\mathcal{V}$ using a family of history-randomized (*HR*) strategies, then one can achieve this via a family of memoryless deterministic (*MD*) strategies.

2. We prove that the almost-sure $\{1,2,3\}$-parity problem for OC-MDPs is at least as hard as the limit-sure selective termination problem for OC-MDPs.

Definition 3.4.1 ([28])**.** A **One-Counter MDP** (OC-MDP) is a tuple $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ where

- Q is a finite set of states which is partitioned into controlled (Q_1) and probabilistic (Q_P) states.
- $\delta^{=0} \subseteq Q \times \{0,1\} \times Q$ is the set of zero rules and $\delta^{=0} \subseteq Q \times \{-1,0,1\} \times Q$ is the set of positive rules, where every $q \in Q$ has an outgoing zero and an outgoing positive rule.
- $P^{=0}$ assigns to every $q \in Q_P$ a positive rational probability distribution over the outgoing transitions in $\delta^{=0}$ of q
- $P^{>0}$ assigns to every $q \in Q_P$ a positive rational probability distribution over the outgoing transitions in $\delta^{>0}$ of q.

Given a OC-MDP, we now define a naturally induced MDP with *boundary* and a *boundaryless* MDP as in Definition 3.4.2 and Definition 3.4.3, respectively.

Definition 3.4.2 (MDP with boundary)**.** A OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ determines an infinite state MDP $\mathcal{M}_{\mathcal{V}} = \langle Q \times \mathbb{N}, Q_1 \times \mathbb{N}, Q_P \times \mathbb{N}, A, \to, p \rangle$ as in the following. For every $p, q \in Q$ with $i \in \mathbb{N}$, we have that

- $A \stackrel{\text{def}}{=} \{-1, 0, 1\}$
- $\langle p, 0 \rangle \xrightarrow{i} \langle q, i \rangle$ if and only if $\langle p, i, q \rangle \in \delta^{=0}$
- If $p \in Q_P$, then the probability of $\langle p, 0 \rangle \xrightarrow{j} \langle q, j \rangle$ is $P^{=0}(p, j, q)$

and for every $p, q \in Q$ with $i \in \mathbb{N}$ and $j \in \mathbb{N}$,

- $\langle p, i \rangle \xrightarrow{j-i} \langle q, j \rangle$ if and only if $\langle p, j-i, q \rangle \in \delta^{>0}$
- If $p \in Q_P$, then the probability of $\langle p, i \rangle \xrightarrow{j-i} \langle q, j \rangle$ is $P^{>0}(p, j-i, q)$

Conversely, we define an MDP with *boundary* as in the following.

Definition 3.4.3 (boundaryless MDP)**.** Every OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ determines an infinite state MDP $\widehat{\mathcal{M}}_{\mathcal{V}} = \langle Q \times \mathbb{Z}, Q_1 \times \mathbb{Z}, Q_P \times \mathbb{Z}, A, \to, p \rangle$ as in the following. For every $p, q \in Q$ with $i, j \in \mathbb{Z}$, we have that

- $A \stackrel{\text{def}}{=} \{-1, 0, 1\}$
- $\langle p, i \rangle \xrightarrow{j-i} \langle q, j \rangle$ if and only if $\langle p, j-i, q \rangle \in \delta^{>0}$
- If $p \in Q_P$, then the probability of $\langle p, i \rangle \xrightarrow{j-i} \langle q, j \rangle$ is $P^{>0}(p, j-i, q)$

A strategy σ on an MDP with boundary or boundaryless MDP is called *counter-oblivious* memoryless-deterministic if there exists a function $h : Q \to \delta^{>0}$ that chooses a transition from each state $q \in Q$ such that at every $\langle q, n \rangle \in Q \times \mathbb{N}$, strategy σ chooses $h(q)$ with probability 1. In this way, the counter value and the history of play is not taken into account.

3.4.1 Objectives for OC-MDPs

We consider *qualitative* objectives for OC-MDPs, such as variants of control-state reachability with imposed conditions on the counter values.

Let $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ be a OC-MDP and $\mathcal{M}_{\mathcal{V}} = \langle C \times \mathbb{N}, C_1 \times \mathbb{N}, C_P \times \mathbb{N}, A, \to, p \rangle$ be an infinite state MDP with boundary as in Definition 3.4.2.

We first define the *termination* objective (denoted as `Term`). Intuitively, it consists of the set of all (infinite) plays of $\mathcal{M}_{\mathcal{V}}$ that encounter a configuration for which the counter

value is zero, regardless of the control state that is visited when this scenario happens. In other words, the objective does not take into account what specific control state is visited, just the fact that the counter value is 0.

Definition 3.4.4 (Termination)**.** Given a OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ with the derived MDP with boundary $\mathcal{M}_{\mathcal{V}} = \langle Q \times \mathbb{N}, Q_1 \times \mathbb{N}, Q_P \times \mathbb{N}, A, \rightarrow, p \rangle$, we define the *termination* objective (denoted as $\texttt{Term}$) as the set of all infinite plays $\rho \stackrel{\text{def}}{=} c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ of $\mathcal{M}_{\mathcal{V}}$ for which there exists an index $i \in \mathbb{N}$ such that $c_i = \langle q, 0 \rangle$, where $q \in Q$.

Given a set of target states $T \subseteq Q$, the *selective termination* objective (denoted as $\texttt{ST}_T$ or for simplicity, just $\texttt{ST}$ if T is understood from the context) consists of the set of all infinite plays of $\mathcal{M}_{\mathcal{V}}$ that encounter a configuration for which the counter value is zero and the control state belongs to T. Clearly, this objective is more restrictive than termination.

Definition 3.4.5 (Selective termination)**.** Given a OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ with the derived MDP with boundary $\mathcal{M}_{\mathcal{V}} = \langle Q \times \mathbb{N}, Q_1 \times \mathbb{N}, Q_P \times \mathbb{N}, A, \rightarrow, p \rangle$ and a subset $T \subseteq Q$ of control states (a.k.a. *target* set), we define the *selective termination* objective (denoted as $\texttt{ST}_T$ and $\texttt{ST}$ if T is understood from the context) as the set of all infinite plays $\rho \stackrel{\text{def}}{=} c_0 \xrightarrow{a_0} c_1 \xrightarrow{a_1} c_2 \cdots$ of $\mathcal{M}_{\mathcal{V}}$ for which there exists an index $i \in \mathbb{N}$ such that $c_i = \langle q, 0 \rangle$, where $q \in T$.

We sometimes will use in our proofs a more expanded notation for termination and selective termination, as in Lemma 3.4.1.

Lemma 3.4.1. Given OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$ with the derived MDP with boundary $\mathcal{M}_{\mathcal{V}} = \langle Q \times \mathbb{N}, Q_1 \times \mathbb{N}, Q_P \times \mathbb{N}, A, \rightarrow, p \rangle$, a target set $T \subseteq Q$, we have that

$$\texttt{ST} = [\![\Diamond (T \times \{0\})]\!]$$

Moreover,

$$\texttt{Term} = [\![\Diamond ((T \cup \neg T) \times \{0\})]\!]$$

Proof. The result follows by unfolding the definitions of the $\texttt{ST}$ and $\texttt{Term}$ objectives, respectively. □

We will define the following two sets on MDPs *with boundary*. We define $\mathit{ValOne}_{\texttt{Term}} \stackrel{\text{def}}{=} \{ \langle p, i \rangle \mid \mathit{Val}(\mathcal{M}_{\mathcal{V}}, \langle p, i \rangle, \texttt{Term}) = 1 \}$ to represent the set of configurations from which

a controller can achieve termination with probability arbitrarily close to 1, i.e., *limit-surely*.

Conversely, we define $OptValOne_{\texttt{Term}} \stackrel{\text{def}}{=} \{\langle p,i\rangle \mid \exists \sigma \in \Pi^{HR}.\mathbb{P}(\mathcal{M}_{\mathcal{V}},\langle p,i\rangle,\sigma,\texttt{Term}) = 1\}$ to represent the set of configurations from which a controller can achieve termination with probability 1, i.e., *almost-surely*.

Theorem 3.4.2 (cf. Theorem 12 of [28])**.** *Given a OC-MDP $\mathcal{V}$, it holds that $ValOne_{\texttt{Term}} = OptValOne_{\texttt{Term}}$. For any configuration $\langle p,i\rangle$ of $\mathcal{V}$, we can decide in polynomial time whether $\langle p,i\rangle \in ValOne_{\texttt{Term}}$. Moreover, there exists a counter-oblivious MD strategy σ which is constructible in polynomial time that is optimal in every configuration of $ValOne_{\texttt{Term}} = OptValOne_{\texttt{Term}}$.*

3.4.2 OC-MDPs

In [28], several algorithmic problems have been treated. In particular, for the *termination* objective, the set of configurations from which one can achieve almost-sure termination coincide with the set of configurations from which one can achieve termination limit-surely. Moreover, deciding whether the optimal probability is 1 has been shown to be in polynomial time, as in Theorem 3.4.2. However, for the *selective termination* objective, the limit-sure case does not coincide with the almost-sure case, i.e., there may not be any optimal strategy, even if the supremum probability of terminating in a desired subset of control state is 1. Note that if from a configuration one can achieve the ST objective almost-surely, then trivially, one can satisfy the ST objective limit-surely as well, but not vice-versa. This fact can be illustrated in Section 3.4.7. In (Theorem 15,[28]) it has been shown that the almost-sure selective termination problem for OC-MDPs is decidable and PSPACE-hard, and provide an exponential time algorithm.

In the case of limit-sure selective termination for OC-MDPs, the problem has shown to be PSPACE-hard [50], but its decidability remains still open. Our main result here is a connection between the limit-sure selective termination and the almost-sure subcase of general parity, called $\{1,2,3\}$-parity problem. Intuitively, every control state is labelled with a number (color) between 1 and 3, needing to have that the maximally color visited infinitely often is 2.

We recall that as stated in the beginning of Chapter 3, the $\{1,2,3\}$-parity is just a small subcase of general parity problem, but already very hard (unlike simpler subcases of parity in the Mostowski hierarchy presented in [61]). Firstly, this is the case because almost-sure $\{1,2,3\}$-parity requires infinite memory on general MDPs (as shown in

[61]). Secondly, even on OC-MDPs, decidability of almost-sure $\{1,2,3\}$-parity is open, and (as shown in this thesis) it is at least as hard as the (also open) limit-sure selective termination problem.

3.4.3 Solvency Games

Solvency games (studied in [20]) model a risk-averse gambler (also known as *investor*). They are a subclass of OC-MDPs, since they have a single control state, but there may exist several actions that can modify the counter value (also known as *bankroll*). Each action (also known as investment choice) is a finitely supported probability distribution on the set of integers. The probability distribution will specify the probabilities for which each payoff (modification in the counter value) is assigned, given a particular action that is chosen. Hence, it is possible that the counter value is modified by more than 1 unit per transition. The objective in solvency games is to minimize the risk of becoming bankrupt, starting with a given strictly positive bankroll. In [20] it has been shown that if the solvency game satisfies certain technical conditions on the eigenvalues of a matrix of a game, there exists a rich man's pure optimal strategy. In other words, once the gambler's bankroll is above a certain threshold, it is optimal to use the same action every time. They compute the optimal strategy under these game restrictions in exponential time. In general however, they show that this optimal strategy may not exist. Moreover, in [28] it has been shown that all qualitative problems for solvency games are decidable in PTIME[2].

3.4.4 Recursive Markov Chains (RMCs) and Recursive Markov Decision Processes (RMDPs)

Recursive Markov Chains (RMCs) denote a class of countably infinite MCs that are constructed by adding a natural recursion feature to finite state MCs.

It has been shown in [42] that adding recursion to stochastic systems provides an abstract model to represent probabilistic procedural programs. A reachability problem for a Recursive Markov Chain is based on calculating the probability for which one can reach a certain control state from the initial one. The work of [29] tackles the reachability and termination problems for RMCs. Namely, they perform both a qualitative and

[2] In other words, for a given solvency game it is decidable in PTIME whether the gambler has a strategy to go bankrupt with probability > 0, $= 0$, or < 1.

quantitative analysis for the reachability and termination objectives and show that these problems can be decided in PSPACE.

RMCs generalize multiple classes of stochastic systems, such as Stochastic Context-Free Grammars (SCFGs) (also known as 1-exit RMCs), Multi-Type Branching Processes [40], Quasi Birth Death Processes (QBDs). It has been shown in [43] that other models such as probabilistic Pushdown Automata (pPDA) [38] and Tree-Like Quasi Birth Death Processes [41] are equivalent to the RMC model.

3.4.5 One-Counter Simple Stochastic Games (OC-SSGs)

One-counter Simple Stochastic Games (OC-SSGs) [26] are a subclass of two-player zero-sum stochastic games played on transition graphs of one-counter automata. The OC-SSGs framework can be considered as a 2-player variant of OC-MDPs, in which some control states belong to another non-deterministic player. Informally, a OC-SSG possesses a finite set of control states, partitioned into three disjoint sets. The first set of control states are under player 1's control (also known as player *Max*), the second set of control state are under player 2's control (also known as player *Min*), whereas the third set of control states are random, i.e under Nature's control. Transitions may change the control state as well as it can decrease/increase by 1 the counter value or it can leave it unchanged. In the case where the set of control states under player *Min*'s control are empty, the system is called a maximizing OC-MDP. Conversely, if there are no control states that belong to player Max, the system is known as a minimizing OC-MDP. Intuitively, player Max would like to *maximize* the probability of achieving a certain objective, whereas player Min would like to minimize it. From the Blackwell's determinacy theorem [65], it follows that objectives such as reachability, termination are *determined*, i.e., they have a value. In the 2-player framework, a value v of a game is represented as the following. For every $\varepsilon > 0$, no matter what strategy player Min uses, player Max has a strategy such that the probability of achieving the objective is $\geq v - \varepsilon$. Also, regardless of what player Max does, player Min has a strategy such that the probability of achieving the objective is $\geq v + \varepsilon$.

For termination objectives, it has been shown in [26] that the value of a OC-SSG can be irrational, even if the system contains rational probabilites on its transitions. This is realized even if the set of Max's and Min's control states are empty, i.e there are only stochastic control states. Moreover, deciding whether the termination value is < 1 is at least as hard as Condon's quantitative reachability problem for Simple Stochastic

Games.

3.4.6 One-Counter Nets (OCNs)

One-Counter Nets (OCNs) consist of a finite control and one integer counter that cannot be tested for zero. In this sense, this model is subsumed by One-Counter Automata, and Pushdown Automata in general, since those allow zero tests by reading a bottom marker on the stack. Moreover, OCNs are a subclass of VASS/Petri nets, being the equivalent of the one-dimensional VASS model or Petri Nets with at most one unbounded place. Hence, multiple decidability questions for VASS apply to OCNs as well. Further details about OCNs and their decidability questions can be found in [57].

3.4.7 Limit-Sure Selective Termination for OC-MDPs

In this subsection, we are interested in the *limit-sure selective termination* problem for OC-MDPs. Given a OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$, a subset $T \subseteq Q$ of *target* states, an initial configuration $\langle p, 1 \rangle \in Q \times \mathbb{N}$, one would like to decide whether it is possible to make the probability of reaching a configuration $\langle q, 0 \rangle \in T \times \mathbb{N}$ *arbitrarily close* to 1.

We now formally state the limit-sure selective termination problem for OC-MDPs.

Limit sure selective termination for OC-MDPs

Input: OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$, $T \subseteq Q$, initial configuration $\langle p, 1 \rangle \in Q \times \mathbb{N}$

Question: Does there exist for every $\varepsilon > 0$, a strategy σ_ε such that $\mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1 \rangle, \sigma_\varepsilon, \mathtt{ST}) \geq 1 - \varepsilon$?

Conversely, we now state the almost-sure $\{1,2,3\}$-parity decision problem on OC-MDPs. Recall from Definition 3.1.11 that the parity objective is defined in terms of a priority (color) function that is given a priori. Given a OC-MDP $\mathcal{V}$ and a function $\lambda : Q \to \{1,2,3\}$, we define a priority $\widehat{\lambda} : Q \times \mathbb{N} \to \{1,2,3\}$ by lifting λ to configurations (states) in the infinite state MDP with boundary $\mathcal{M}_\mathcal{V}$.

Almost sure $\{1,2,3\}$-PAR problem for OC-MDPs

Input: OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0} \rangle$, initial configuration $\langle q, 1 \rangle \in Q \times \mathbb{N}$, a function $\lambda : Q \to \{1,2,3\}$

Question: Does there exist a strategy $\sigma \in \Pi$ such that $\mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle q, 1 \rangle, \sigma, \{1,2,3\}\text{-PAR}(\widehat{\lambda})) = 1$?

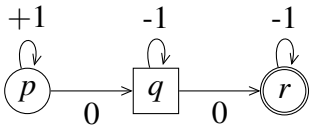


Figure 3.7: An illustration of the limit sure selective termination problem for a OCMDP $\mathcal{V}$. The circles states are controlled, whereas the square states are probabilistic. The target control state r is drawn with double edges. From $\langle p, 1 \rangle$, the configuration $\langle r, 0 \rangle$ can be reached limit-surely (by a family of strategies that repeats the loop at p more and more often), but not almost-surely, since every strategy has a non-zero chance of getting stuck at state q with counter value zero. Each probabilistic transition has chance $\frac{1}{2}$

In Figure 3.7, similar as in [28], we present an illustration of the limit-sure selective termination problem for a OC-MDP $\mathcal{V}$. It is not hard to observe that for all $\varepsilon > 0$, there exists a strategy σ_ε, for every configuration $\langle p, i \rangle \in Q \times \mathbb{N}$, such that $\mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle p, i \rangle, \sigma_\varepsilon, \text{ST}) \geq 1 - \varepsilon$. The intuition behind this construction is the fact that for every $\varepsilon > 0$, one can construct a strategy σ_ε in the following way. Starting at configuration $\langle p, 1 \rangle$, we define $k_\varepsilon \in \mathbb{N}$ (which is chosen in terms of the value of ε) so that the controller can "pump the counter up" by using the transition $\langle p, +1, p \rangle$ k_ε many times. Then, from configuration $\langle p, k_\varepsilon \rangle$ move to state q by using the transition $\langle p, 0, q \rangle$. Then, there exists a chance of $\frac{1}{2}$ of staying in q and chance of $\frac{1}{2}$ of going to target control state r. It is easy to observe that each strategy σ_ε reaches configuration $\langle r, 0 \rangle$ with probability $\mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle p, 1 \rangle, \sigma_\varepsilon, \text{ST}) \geq 1 - \frac{1}{2^{k_\varepsilon}} > 1 - \varepsilon$. However, there is no chance to reach state r with counter value 0 with probability 1, since there is a strictly positive chance of getting stuck at control state q.

Furthermore, for this particular example, the constructed strategies are both memoryless deterministic (MD) and finitely representable by a deterministic finite automaton. A further analysis concerning this fact is presented in [28].

Lemma 3.4.3 (cf. Theorem B, [74])**.** Given a countable state MDP $\mathcal{M} = \langle C, C_1,$

$, C_P, A, \to, p\rangle$ and a set $Q \subseteq C$, then for every $\varepsilon > 0$, there exists a *memoryless-deterministic* $\sigma_\varepsilon \in \Pi^{MD}$ such that for every $c \in C$,

$$\mathbb{P}(\mathcal{M}, c, \sigma_\varepsilon, [\![\Diamond Q]\!]) \geq (1-\varepsilon) \times \sup_{\sigma \in \Pi^{HR}} \mathbb{P}(\mathcal{M}, c, \sigma, [\![\Diamond Q]\!])$$

We prove in Theorem 3.4.4 that in order to achieve limit sure selective termination on OC-MDPs, it suffices to use a memoryless-deterministic strategy only.

Theorem 3.4.4. *Given a OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}\rangle$, and the derived MDP with boundary $\mathcal{M}_\mathcal{V} = \langle Q \times \mathbb{N}, Q_1 \times \mathbb{N}, Q_P \times \mathbb{N}, A, \to, p\rangle$, a set of target states $T \subseteq Q$ and initial control state $\langle p, 1\rangle$, we have that limit sure selective termination can be achieved if and only if limit sure selective termination can be achieved using a family of MD strategies, i.e.*

$$\sup_{\sigma \in \Pi^{HR}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, \mathtt{ST}) = 1 \iff \sup_{\sigma \in \Pi^{MD}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, \mathtt{ST}) = 1$$

Proof. Case $\Longleftarrow$ is trivial since by [76], any memoryless deterministic strategy is also a history randomized one, i.e. $\Pi^{MD} \subseteq \Pi^{HR}$. Hence, the result follows.

Case $\Longrightarrow$. Assume that $\sup_{\sigma \in \Pi^{HR}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, \mathtt{ST}) = 1$. By Lemma 3.4.3 (let $Q \stackrel{\text{def}}{=} T \times \{0\}$), we have that for every $\varepsilon > 0$, there exists $\sigma_\varepsilon \in \Pi^{MD}$ such that

$$\mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma_\varepsilon, [\![\Diamond T \times \{0\}]\!]) \geq (1-\varepsilon) \times \sup_{\sigma \in \Pi^{HR}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, [\![\Diamond T \times \{0\}]\!])$$

From Lemma 3.4.1, we have that $\mathtt{ST} = [\![\Diamond T \times \{0\}]\!]$. Hence, by our assumption, since $\sup_{\sigma \in \Pi^{HR}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, \mathtt{ST}) = 1$, we obtain that

$$\forall \varepsilon > 0. \exists \sigma_\varepsilon \in \Pi^{MD}. \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma_\varepsilon, \mathtt{ST}) \geq (1-\varepsilon)$$

and so that $\sup_{\sigma \in \Pi^{MD}} \mathbb{P}(\mathcal{M}_\mathcal{V}, \langle p, 1\rangle, \sigma, \mathtt{ST}) = 1$.

□

Now we prove that the almost-sure $\{1,2,3\}$ parity problem for OC-MDPs is at least as hard as the limit sure selective termination problem for OC-MDPs. This result is presented in Theorem 3.4.7.

The main idea of the reduction is the following. From the OC-MDP $\mathcal{V}$ with control states Q, an initial control state p and a target set $T \subseteq Q$, we construct a OC-MDP $\mathcal{V}'$ with control states Q' and a priority function $\lambda' : Q' \to \{1,2,3\}$, by keeping the control states of $\mathcal{V}$ and add two new probabilistic control states t and f. The transition rules from $\mathcal{V}$ are kept, along with some new zero rules. In particular, from every control state

in T where the counter value is zero, there exists a zero rule (taken with probability 1) towards control state t. Conversely, from every control state not in T where the counter value is zero, there exists a zero rule towards control state f. Furthermore, from both t and f there exists a zero rule towards initial control state p which increments the counter by 1. The priority function λ' labels control state t and f with color 2, and 3, respectively, whereas every other control state is labelled by color 1.

Hence, one can achieve limit-sure selective termination in $\mathcal{V}$ if and only if one achieve almost-sure $\{1,2,3\}$-parity in $\mathcal{V}'$.

Definition 3.4.6. Given a OC-MDP $\mathcal{V} = \langle Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0}\rangle$ with target set $T \subseteq Q$, an initial control state $p_i \in Q$, we construct a OC-MDP $\mathcal{V}' = \langle Q', Q'_1, Q'_P, \delta'^{=0}, \delta'^{>0}, P'^{=0}, P'^{>0}\rangle$ and a priority function $\lambda' : Q' \to \{1,2,3\}$ as

- $Q'_1 \stackrel{\text{def}}{=} Q_1$; $Q'_P \stackrel{\text{def}}{=} Q_P \cup \{t,f\}$
- $\delta'^{>0} \stackrel{\text{def}}{=} \delta'^{>0}$
- $\delta'^{=0} \stackrel{\text{def}}{=} \delta^{=0} \cup \{(p,0,t) \mid p \in T\} \cup \{(p,0,f) \mid p \notin T\} \cup \{\langle t,+1,p_i\rangle, \langle f,+1,p_i\rangle\}$
- $P'^{=0}(\langle x,op,y\rangle) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \{x,y\} \cap \{t,f\} \neq \varnothing \\ P^{=0}(\langle x,op,y\rangle) & \text{if } x \in Q_P \wedge y \in Q \\ 0 & \text{if } \textit{otherwise} \end{cases}$
- $\lambda'(q) = 1$, for all $q \notin \{t,f\}$; $\lambda'(t) = 2$; $\lambda'(f) = 3$

We state the Borel-Cantelli lemma [33], which will be used in the proof of Theorem 3.4.7

Lemma 3.4.5 (Borel-Cantelli lemma)**.** [33] Let $(E_n)_{n\in\mathbb{N}}$ be a sequence of events in a probability space. Let us denote the event $E^\infty \stackrel{\text{def}}{=} \bigcap_{k=1}^{\infty} \bigcup_{n=k}^{\infty} E_n$, which denotes the fact that E_n occurs for infinitely many n. If $\Sigma_{n=1}^{\infty} \mathbb{P}(E_n) < \infty$ then $\mathbb{P}(E^\infty) = 0$.

We will show that if limit-sure selective termination holds from an initial configuration on a OCMDP $\mathcal{V}$, then limit-sure selective termination holds by always remaining in a set of configurations from which one would almost-surely reach a configuration $\langle q,0\rangle$, where $q \notin T$.

Lemma 3.4.6. Given a OC-MDP $\mathcal{V}$ with states Q, an initial configuration $c \in Q \times \mathbb{N}$ and a set of target states $T \subseteq Q$, we have that

$$Val(\mathcal{M}_\mathcal{V}, c, \mathtt{ST}) = 1 \implies Val(\mathcal{M}_\mathcal{V}, c, \mathtt{ST} \cap [\![\Box C]\!]) = 1$$

where $C \stackrel{\text{def}}{=} \{c' \in Q \times \mathbb{N} \mid Val(\mathcal{M}_\mathcal{V}, c', \mathtt{Term}) = 1\}$.

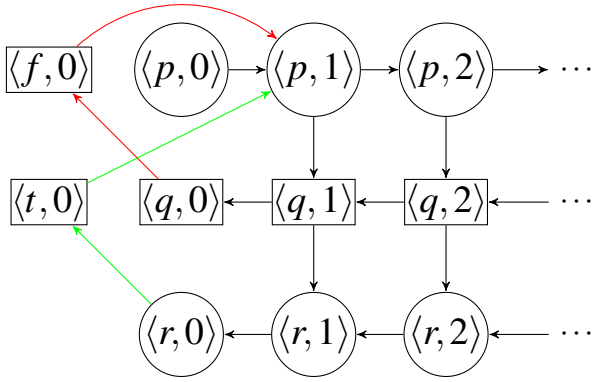


Figure 3.8: An infinite state MDP $\mathcal{M}_{\mathcal{V}'}$ derived from OC-MDP $\mathcal{V}'$ (which is itself induced by the OCMDP $\mathcal{V}$ from Figure 3.7) as in Definition 3.4.6. The non-deterministic states are represented as circles and probabilistic states drawn as squares. Configurations $\langle t,0\rangle$ and $\langle f,0\rangle$ are labelled with colors 2 and 3, respectively, whereas all other configurations have priority 1. Given a probabilistic state $s \in Q_P \times \mathbb{N}_0$, every transition is taken with probability $\frac{1}{2}$. From $s \in \{\langle t,0\rangle, \langle f,0\rangle\}$ there is a unique transition to $\langle p,1\rangle$ (illustrated by the green and red arrows, respectively) that is taken with probability 1.

Proof. By Theorem 3.4.2, it holds that the set of configurations of C satisfy $\texttt{Term}$ objective almost-surely, using a counter-oblivious memoryless strategy σ. Moreover, for every $\langle q,k\rangle \in Q \times \mathbb{N}$, we have that $Val(\mathcal{M}_{\mathcal{V}}, \langle q,k+1\rangle, \texttt{Term}) \leq Val(\mathcal{M}_{\mathcal{V}}, \langle q,k\rangle, \texttt{Term})$, as stated in [27]. In other words, for any control state q, the value of achieving termination decreases as the counter value increases. For any control state $q \in Q$, let us consider the smallest $k \in \mathbb{N}$ such that $Val(\mathcal{M}_{\mathcal{V}}, \langle q,k\rangle, \texttt{Term}) < 1$, and define $v_q \stackrel{\text{def}}{=} Val(\mathcal{M}_{\mathcal{V}}, \langle q,k\rangle, \texttt{Term})$, if such k exists. Otherwise, disregard the control state q. Since Q is finite, there exists a minimum value for v_q and we define $\delta \stackrel{\text{def}}{=} \min_q(1-v_q)$. Note that by construction, $\delta > 0$. Given $\varepsilon > 0$, let us fix $\sigma_\varepsilon \in \Pi^{HR}$ such that $\mathbb{P}(\mathcal{M}_{\mathcal{V}}, c, \sigma_\varepsilon, \texttt{ST}) \geq 1-\varepsilon$. Let us define $\gamma \stackrel{\text{def}}{=} \mathbb{P}(\mathcal{M}_{\mathcal{V}}, c, \sigma_\varepsilon, [\![\neg\Box C]\!])$, namely the probability that σ_ε eventually reaches a configuration in $(Q \times \mathbb{N}) \setminus C$, which does not satisfy the $\texttt{Term}$ objective almost surely. We have that

$$\mathbb{P}(\mathcal{M}_{\mathcal{V}}, c, \sigma_\varepsilon, \neg\texttt{ST}) \geq \gamma \times \delta \tag{3.2}$$

Since the $\texttt{ST}$ objective is satisfied limit-surely, we have that

$$\mathbb{P}(\mathcal{M}_{\mathcal{V}}, c, \sigma_\varepsilon, \neg\texttt{ST}) < \varepsilon \tag{3.3}$$

From Equation (3.2) and Equation (3.3), we obtain $\gamma \leq \frac{\varepsilon}{\delta}$.

Let us pick an arbitrary strategy σ'_ε that replicates the same moves of σ_ε up until a

configuration in $(Q \times \mathbb{N}) \setminus C$ is reached, and fail otherwise. Namely, for all plays ρ with $\sigma_\varepsilon(\rho) \notin (Q \times \mathbb{N}) \setminus C$, it holds that $\sigma'_\varepsilon(\rho) = \sigma_\varepsilon(\rho)$. Using $\gamma \leq \frac{\varepsilon}{\delta}$, we obtain:

$$\begin{aligned} \mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle p, 1\rangle, \sigma'_\varepsilon, \mathtt{ST}) &\geq 1 - \varepsilon - \gamma \\ &\geq 1 - \varepsilon - \frac{\varepsilon}{\delta} \\ &= 1 - \varepsilon \left(1 - \frac{1}{\delta}\right) \end{aligned} \tag{3.4}$$

Since $(1 - \frac{1}{\delta})$ is independent of ε, one can still satisfy the ST objective with probability arbitrarily close to 1, by never visiting a configuration that does not terminate with probability 1, i.e. always remain in the set C.

□

We will now provide a polynomial time reduction from the limit sure selective termination for OC-MDPs to the almost sure $\{1,2,3\}$-parity for OC-MDPs.

Theorem 3.4.7. *Given $\mathcal{V} = (Q, Q_1, Q_P, \delta^{=0}, \delta^{>0}, P^{=0}, P^{>0})$ be a OC-MDP with initial configuration $\langle p, 1\rangle$ and a set of target states $T \subseteq Q$, one can construct in logarithmic space (and thus in polynomial time) a OC-MDP $\mathcal{V}'$ with priority function $\lambda' : Q' \to \{1,2,3\}$, as in Definition 3.4.6. Then, limit-sure selective termination holds in $\mathcal{V}$ if and only if almost sure $\{1,2,3\}$-parity holds in $\mathcal{V}'$.*

$$\forall \varepsilon > 0 \exists \sigma_\varepsilon \in \Pi^{HR}. \mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle p, 1\rangle, \sigma_\varepsilon, \mathtt{ST}) \geq 1 - \varepsilon \iff$$

$$\exists \sigma \in \Pi^{HR}. \mathbb{P}(\mathcal{M}_{\mathcal{V}'}, \langle p, 1\rangle, \sigma, \{1,2,3\} - \mathtt{PAR}(\widehat{\lambda'})) = 1$$

Proof. Case $\Longleftarrow$. Assume that limit sure selective termination does not hold in $\mathcal{M}_{\mathcal{V}}$. Hence, there must exist $\varepsilon > 0$ such that

$$\forall \sigma \in \Pi^{HR}. \mathbb{P}(\mathcal{M}_{\mathcal{V}}, \langle p, 1\rangle, \sigma, \mathtt{ST}) < 1 - \varepsilon \tag{3.5}$$

We show that there is no strategy $\sigma' \in \Pi^{HR}$ such that $\mathbb{P}(\mathcal{M}_{\mathcal{V}'}, \langle p, 1\rangle, \sigma, \{1,2,3\}\text{-}\mathtt{PAR}(\widehat{\lambda'})) = 1$. We will prove this fact by contradiction.

Let us assume that almost sure $\{1,2,3\}$-parity holds in $\mathcal{M}_{\mathcal{V}'}$, i.e.,

$$\exists \sigma' \in \Pi^{HR}. \mathbb{P}(\mathcal{M}_{\mathcal{V}'}, \langle p, 1\rangle, \sigma', \{1,2,3\} - \mathtt{PAR}(\widehat{\lambda'})) = 1 \tag{3.6}$$

From the hypothesis and by construction of $\mathcal{M}_{\mathcal{V}'}$, w.l.o.g.,

$$\exists \varepsilon > 0. \forall \tau \in \Pi^{HR}. \mathbb{P}(\mathcal{M}_{\mathcal{V}'}, \langle p, 1\rangle, \tau, [\![\Diamond(\langle t, 0\rangle)]\!]) < 1 - \varepsilon < 1 \tag{3.7}$$

However, from the definition of the $\{1,2,3\}$-parity objective we have that $\{1,2,3\}(\widehat{\lambda'}) \subseteq [\![\Box\Diamond(\langle t,0\rangle)]\!]$. Therefore, by Equation (3.7), we obtain the following inequality

$$\forall \tau \in \Pi^{HR}.\mathbb{P}(\mathcal{M}_{\mathcal{V}'},\langle p,1\rangle,\tau,\{1,2,3\}-\mathtt{PAR}(\widehat{\lambda'})) \leq \mathbb{P}(\mathcal{M}_{\mathcal{V}'},\langle p,1\rangle,\tau,[\![\Diamond(\langle t,0\rangle)]\!]) < 1 \tag{3.8}$$

which contradicts Equation (3.6). Hence, there is no strategy on $\mathcal{M}_{\mathcal{V}'}$ that satisfies the $\{1,2,3\}$-parity objective almost-surely.

Case $\Longrightarrow$. Assume that limit-sure selective termination holds in $\mathcal{M}_{\mathcal{V}}$, i.e.,

$$\forall \varepsilon > 0 \exists \sigma_\varepsilon \in \Pi^{HR}.\mathbb{P}(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}) \geq 1-\varepsilon.$$

Given $\varepsilon > 0$, fix strategy σ_ε on $\mathcal{M}_{\mathcal{V}}$ such that the ST objective is satisfied with probability $\geq 1-\varepsilon$. We define $\mathbb{P}_{k_\varepsilon}(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST})$ to be the probability that the ST objective is satisfied within the first k steps. Observe that

$$\mathbb{P}_{k_\varepsilon}(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}) \leq \mathbb{P}(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}).$$

and

$$\lim_{k\to\infty} \mathbb{P}_k(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}) = \mathbb{P}(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}) \geq 1-\varepsilon.$$

Therefore, for every $\varepsilon > 0$, there exists a number $k_\varepsilon \in \mathbb{N}$ such that $\mathbb{P}_k(\mathcal{M}_{\mathcal{V}},\langle p,1\rangle,\sigma_\varepsilon,\mathtt{ST}) \geq 1-2\varepsilon$.

Now we construct a strategy τ on $\mathcal{M}_{\mathcal{V}'}$ as in the following. For every $\varepsilon_i > 0$, replicate the same moves as σ_{ε_i} for the first k_{ε_i} steps. By construction, $\mathbb{P}_{k_{\varepsilon_i}}(\mathcal{N},\langle p,1\rangle,\sigma_{\varepsilon_i},[\![\Diamond\langle t,0\rangle]\!] \geq 1-2\varepsilon_i$. Then, let us denote $c_{k_{\varepsilon_i}}$ be the configuration at step k_{ε_i}. By Lemma 3.4.6 we can assume without restriction that $c_{k_{\varepsilon_i}} \in C$, where $C \stackrel{\text{def}}{=} \{c' \in Q\times\mathbb{N} \mid Val(\mathcal{M}_{\mathcal{V}},c',\mathtt{Term}) = 1\}$, and then switch to an existing strategy σ to ensure that $\mathbb{P}(\mathcal{M}_{\mathcal{V}'},c_k,\sigma,[\![\Diamond(\langle t,0\rangle \vee \langle f,0\rangle)]\!]) = 1$. By construction, the system is guaranteed to restart in $\langle p,1\rangle$ and then the strategy τ restarts the process with a smaller ε_i (see below).

Now we argue that $\mathbb{P}(\mathcal{M}_{\mathcal{V}'},\langle p,1\rangle,\tau,[\![\Box(\Diamond\langle f,0\rangle)]\!]) = 0$. For simplicity, let us define $S \stackrel{\text{def}}{=} Q\times\mathbb{N}_0$. We define the sequence of events E_l of visiting $\langle f,0\rangle$ between the l-th and $(l+1)$-th visits of $\langle p,1\rangle$, i.e.,

$$E_l \stackrel{\text{def}}{=} (\langle p,1\rangle(S\setminus\{\langle p,1\rangle\})^*)^{l-1}\langle p,1\rangle(S\setminus\{\langle p,1\rangle,\langle f,0\rangle\})^*\langle f,0\rangle\langle p,1\rangle S^\omega$$

We apply the Borel-Cantelli lemma [33] to show that infinitely many of events E_l occur with probability zero. Let $\varepsilon_i \stackrel{\text{def}}{=} 2^{-(i+1)}$. We have that

$$\Sigma_{l=1}^{\infty}\mathbb{P}(\mathcal{M}_{\mathcal{V}'},\langle p,1\rangle,\tau,E_l) = 1+\frac{1}{2}+\ldots+\frac{1}{2^{i+1}}+\ldots = 2 < \infty$$

Using the Borel-Cantelli lemma we now have that $\mathbb{P}(\mathcal{M}_{\mathcal{V}''}, \langle p, 1\rangle, \tau, [\![\Box(\Diamond\langle f, 0\rangle)]\!]) = 0$ and hence, $\mathbb{P}(\mathcal{M}_{\mathcal{V}''}, \langle p, 1\rangle, \tau, \{1,2,3\}\text{-PAR}) = 1$.

□

Chapter 4

Conclusion and Outlook

We studied decidability and complexity questions for timed and probabilistic extensions of Petri nets.

In the first part of the thesis (Chapter 2), we have shown that the *Existential Coverability* problem (and its dual of universal safety) is PSPACE-complete for a timed extension model of Petri nets, called Timed Petri Nets. Our model corresponds to a controller-less timed network where each process is a 1-clock Timed Automata, interacting via handshake communication. The corresponding problem for a Timed Network *with* a central controller is complete for $F_{\omega^{\omega}}$ [55]. In the Timed Petri Net model, every token has a real-valued clock (a.k.a age), and transition firing is constrained by the clock values that have integer bounds (using strict and non-strict inequalities). The newly created tokens can either inherit the age from an input token of the transition or it can be reset to zero. We hence positively solve an open question from [6] concerning the decidability of universal safety in timed network with no central controller. Furthermore, we can compute a symbolic representation of the set of markings which are coverable, using exponential space (Theorem 2.7.14). We show the PSPACE lower bound (Section 2.6) by a reduction from the iterated monotone Boolean circuit problem. Note however that this result does not follow directly from the PSPACE-completeness of the reachability problem for timed automata [12] due to the absence of the global controller. In order to show the PSPACE upper bound, we provide a logspace reduction of the Existential Coverability problem for Timed Petri Nets to the corresponding problem for a syntactic subclass, called *non-consuming* Timed Petri Net (Lemma 2.7.1). We then perform an abstraction of the real-valued clocks, similar to the one used in [8]. Clock values are split into integer parts and fractional parts. The integer parts of the clocks can be abstracted Section 2.7.3 into a finite domain, since the transition

guards cannot distinguish between values above the maximal constant that appears in the system. The fractional parts of the clock values that occur in a marking are ordered sequentially. Then every marking can be abstracted into a string where all the tokens with the i-th fractional clock value are encoded in the i-th symbol in the string. Since token multiplicities do not matter for Existential Coverability, the alphabet from which these strings are built is finite. The primary difficulty is that the length of these strings can grow dynamically as the system evolves, i.e., the space of these strings is still infinite for a given Timed Petri Net. We perform a forward exploration of the space of reachable strings. By using an acceleration technique (Algorithm 1), we can effectively construct a symbolic representation of the set of reachable strings in terms of finitely many regular expressions. Finally, we can check Existential Coverability by using this symbolic representation (Theorem 2.7.14).

It remains an open question whether these positive results for the controller-less case of timed network model can be generalized to multiple real-valued clocks per token. This problem has been considered in this project as well but several issues occurred when reasoning about how clock values relate to each other in a similar acceleration technique as presented here for the one-clock case. In the case *with* a controller, safety becomes undecidable already for two clocks per token [6].

Another question is whether our results can be extended to more general versions of Timed Petri Nets. In our version, clock values are either inherited, advanced as time passes, or reset to zero. However, other versions of Timed Petri Nets allow the creation of output-tokens with new non-deterministically chosen non-zero clock values, e.g., the timed Petri nets of [8, 10] and the read-arc timed Petri nets of [24].

In the second part of the thesis (Chapter 3), we referred to systems with controlled behaviour that are probabilistic extensions of Vector Addition Systems with States (VASS) and One-Counter Automata. We studied the decidability of probability-1 qualitative qualitative reachability and Büchi objectives for infinite-state Markov Decision Processes (MDPs) that are induced by probabilistic extensions of VASS called VASS-MDPs. Several quantitative objectives in probabilistic VASS are either undecidable, or the solution is at least not effectively expressible in $(\mathbb{R}, +, *, \leq)$ [7]. For general VASS-MDPs, we show that even the simplest of these problems, such as (almost)-sure reachability, is undecidable (see Section 3.2.3). We consider two monotone subclasses of VASS-MDPs: 1-VASS-MDPs and P-VASS-MDPs. These are called *single-sided*, since either the controller or the probabilistic player can change counter values. In 1-VASS-MDPs, only Player 1 can modify counter values while the probabilistic player

can only change control-states, whereas for P-VASS-MDPs it is vice-versa. These two models induce infinite-state MDPs as well. We show that the limit-sure control state reachability problem in 1-VASS-MDPs is decidable (Theorem 3.3.5). For our decidability result, we use an algorithm which at each iteration reduces the dimension of the considered 1-VASS-MDP while preserving the limit-sure reachability properties. The limit-sure repeated reachability for 1-VASS-MDPs has been still left open, since several difficulties may arise. In particular, a solution might involve an analysis of the long run behaviour of multi-dimensional random walks induced by probabilistic VASS. In [30] (Section 5) it has been shown that this may exhibit strange non-regular behaviours where the dimension is ≥ 3.

One counter Markov decision processes (OC-MDPs) are a class of infinite state MDPs that are generated by finite-state automata that possess a single unbounded counter. We show in Theorem 3.4.4 that for the limit-sure selective termination objective on OC-MDPs, memoryless-deterministic strategies are *sufficient*. In other words, for a OC-MDP $\mathcal{V}$, if one can achieve limit-sure selective termination on $\mathcal{V}$, then one can achieve this via a family of memoryless-deterministic strategies. Furthermore, we prove in Theorem 3.4.7 that the almost-sure $\{1,2,3\}$-parity problem for OC-MDPs is at least as hard as the limit-sure selective termination problem for OC-MDPs. Note that Theorem 3.4.7 relates a limit-sure problem with an almost-sure problem (for OC-MDPs). In this project, the decidability problem for limit-sure selective termination for OC-MDPs has been considered as well, but several issues have been encountered, when reasoning about limit-sure strategies. Nevertheless, the decidability of both $\{1,2,3\}$-parity problem and limit-sure selective termination problem for OC-MDPs is still open.

Bibliography

[1] P. Abdulla, R. Ciobanu, R. Mayr, A. Sangnier, and J. Sproston. Qualitative analysis on VASS-induced MDPs. In *International Conference on Foundations of Software Science and Computational Structures (FoSSaCS)*, volume 9634 of *LNCS*, pages 319–334. Springer, 2016.

[2] P. Abdulla, G. Delzano, and A. Rezine. Monotonic abstraction in parameterized verification. In *Electronic Notes in Theoretical Computer Science*, volume 223, pages 3–14, 2008.

[3] P. Abdulla and B. Jonsson. Model checking of systems with many identical processes. *Theoretical Computer Science*, 290(1):241–264, 2003.

[4] P. Abdulla and R. Mayr. Priced timed Petri nets. *Logical Methods in Computer Science*, 9(4), 2013.

[5] P. A. Abdulla, R. Ciobanu, R. Mayr, A. Sangnier, and J. Sproston. Universal safety for timed Petri nets is PSPACE-complete. In *International Conference of Concurrency theory (CONCUR)*. LIPIcs (DOI: 10.4230/LIPIcs.CONCUR.2018.6), 2018.

[6] P. A. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 345–354, 2004.

[7] P. A. Abdulla, N. B. Henda, and R. Mayr. Decisive Markov Chains. *Logical Methods in Computer Science*, 3(4), 2007.

[8] P. A. Abdulla, P. Mahata, and R. Mayr. Dense-timed Petri nets: Checking Zenoness, token liveness and boundedness. *Logical Methods in Computer Science*, 3(1), 2007.

[9] P. A. Abdulla, R. Mayr, A. Sangnier, and J. Sproston. Solving parity games on integer vectors. In *International Conference of Concurrency theory (CONCUR)*, volume 8052 of *LNCS*, pages 106–120. Springer, 2013.

[10] P. A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2001.

[11] R. Alur. Timed automata. *Theoretical Computer Science*, 126:183–235, 1999.

[12] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[13] R. Alur and P. Madhusudan. Decision problems for timed automata. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT, Revised Lectures*, volume 3185 of *LNCS*, pages 1–24. Springer, 2004.

[14] S. Asmussen. *Applied Probability and Queues*. Springer, 1987.

[15] F. Avellaneda and R. Morin. Vector Addition Systems with States vs. Petri nets. *Theoretical Computer Science*, 2012.

[16] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[17] J. Beck and N. Wilson. Proactive algorithms for job show scheduling with probabilistic durations. *Journal of Artificial Intelligence Research*, 28:183–232, 2007.

[18] G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Systems (revised lectures)*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–237, 2004.

[19] G. Behrmann, K. Larsen, and J. I. Rasmussen. Priced timed automata: Decidability results, algorithms and applications. In *Proc. 3rd International Symposium on Formal Methods for Components and Objects (FMCO '04)*, volume 3657 of *LNCS*, pages 162–186, 2004.

[20] N. Berger, N. Kapur, L. J. Schulman, and V. Vazirani. Solvency games. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 61–72, 2008.

[21] R. Bonnet, A. Finkel, S. Haddad, and F. Rosa-Velardo. Ordinal theory for the expresiveness of Well Structured Transition Systems. In *Information and Computation*, volume 224, pages 1–22, 2013.

[22] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In *International Conference on Computer-Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531. Springer, 1998.

[23] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model checking. In *International Conference of Concurrency theory (CONCUR)*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[24] P. Bouyer, S. Haddad, and P.-A. Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 420–431. Springer, 2006.

[25] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4052 of *Lecture Notes in Computer Science*, pages 577–588. Springer, 2006.

[26] T. Brazdil, V. Brozek, and K. Etessami. One-counter stochastic games. In *Proc. of Ann. Conf. on Foundations of Software Technology and theoretical computer science (FSTTCS'10)*, pages 108–119, 2010.

[27] T. Brazdil, V. Brozek, K. Etessami, and A. Kucera. Approximating the termination value of one-counter MDPs and stochastic games. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 332–343, 2011.

[28] T. Brazdil, V. Brozek, K. Etessami, A. Kucera, and D. Wojtczak. One-counter Markov Decision Processes. In *Proceedings of 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 863–874, 2010.

[29] T. Brazdil, V. Brozek, V. Forejt, and A. Kucera. Reachability in Recursive Markov Decision Processes. In *International Conference of Concurrency theory (CONCUR)*, pages 358–374, 2006.

[30] T. Brázdil, S. Kiefer, A. Kučera, and P. Novotný. Long-run average behaviour of probabilistic Vector Addition Systems. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 44–55, 2015.

[31] K. Chatterjee and L. Doyen. Energy and mean payoff parity Markov Decision Processes. In *Mathematical Foundations of Computer Science*, volume 7119 of *Lecture Notes in Computer Science*, pages 37–46. Springer, 2011.

[32] K. Chatterjee, M. Jurdziński, and T. Henzinger. Simple stochastic parity games. In *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 100–113, 2003.

[33] K. L. Chung. *A course in probability theory*. Academic Press, 3rd edition, 2001.

[34] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, Feb. 1992.

[35] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. In *American Journal of Mathematics*, volume 34, pages 413–422, 1913.

[36] V. Eberhard. *A first course in systems biology*. Garland Science, 2012.

[37] D. d. F. Escrig, V. V. Ruiz, and O. M. Alonso. Decidability of properties of timed-arc Petri nets. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 1825 of *Lecture Notes in Computer Science*, pages 187–206. Springer, 2000.

[38] J. Esparza, A. Kucera, and R. Mayr. Model checking probabilistic pushdown automata. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 12–21, 2004.

[39] J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Journal of Information Processing and Cybernetics*, 30(3):210–242, 1994.

[40] K. Etessami, A. Stewart, and M. Yannakakis. Polynomial-time algorithms for multi-type branching processes and stochastic context-free grammars. In *Symposium on Theory of Computing Conference (STOC)*, pages 579–588, 2012.

[41] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-Birth Death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. In

Proc. 5th International Symposium on Quantitative Evaluation of Systems (QEST), pages 243–253, 2008.

[42] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *Proc. 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, pages 253–270, 2005.

[43] K. Etessami and M. Yannakakis. Recursive Markov Chains, stochastic grammars, and monotone systems of nonlinear equations. In V. Diekert and B. Durand, editors, *22nd Annual Symposium on the Theoretical Aspects of Computer Science (STACS '05)*, pages 340–352, 2005.

[44] K. Etessami and M. Yannakakis. Recursive Markov Decision Processes and recursive stochastic games. In *Proc. of ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 891–903, 2005.

[45] K. Etessami and M. Yannakakis. Efficient qualitative analysis of classes of recursive Markov Decision Processes and simple stochastic games. In *Proc. of 23rd STACS'06*, Springer, pages 634–645, 2006.

[46] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

[47] A. Finkel and P. Schnoebelen. Fundamental structures in well-structured infinite transition systems. In *Proc. LATIN '98*, volume 1380 of *LNCS*, pages 102–118, 1998.

[48] H. Gimbert. Pure stationary optimal strategies in Markov Decision Processes. In *STACS*, pages 200–211, 2007.

[49] E. Goles, P. Montealegre, V. Salo, and I. Trm. PSPACE-completeness of majority automata networks. *Theoretical Computer Science*, 609(1):118 – 128, 2016.

[50] S. Göller and M. Lohrey. Branching-time model checking of one-counter processes. *SIAM Journal of Computation*, 42(3):884–923, 2013.

[51] S. Göller, R. Mayr, and A. W. Lin. On the computational complexity of verifying One-Counter Processes. *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*, pages 235–244, 2009.

[52] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley and Sons, 1998.

[53] P. Habermehl. On the complexity of the linear-time mu calculus for Petri nets. In *ICATPN '97*, volume 1248 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 1997.

[54] M. H. T. Hack. Decidability questions for Petri nets. *PhD Thesis, MIT*, 1976.

[55] S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal recursive complexity of timed-arc Petri nets, Data nets, and other enriched nets. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 355–364, 2012.

[56] J. Hale. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):609–642, 2006.

[57] P. Hofman, S. Lasota, R. Mayr, and P. Totzke. Simulation problems over one-counter nets. *Logical Methods in Computer Science*, 12(1), 2016.

[58] J. Hopcroft and J. J. Pansiot. On the reachability problem for 5-dimensional Vector Addition Systems. *Theoretical Computer Science*, 8(2):135–159, 1979.

[59] L. Jacobsen, M. Jacobsen, M. H. Møller, and J. Srba. Verification of timed-arc Petri nets. In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 6543 of *Lecture Notes in Computer Science*, pages 46–72, 2011.

[60] R. Karp and R. Miller. Parallel program schemata. *Journal of Computer and system Science*, 3(2):147–195, 1969.

[61] S. Kiefer, R. Mayr, M. Shirmohammadi, and D. Wojtczak. Parity objectives in countable MDPs. In *32th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 1–11, 2017.

[62] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[63] R. Lazić, T. Newcomb, J. Ouaknine, A. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274, 2008.

[64] R. J. Lipton. The reachability problem requires exponential space. Technical report, Yale University, Department of Computer Science, 1976.

[65] D. A. Martin. The determinacy of blackwell games. *Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

[66] A. Mas-Collel, M. D. Whinston, and J. Green. *Microeconomic theory*. Oxford University Press, 1995.

[67] E. W. Mayr. An algorithm for the general Petri net reachability problem. In *STOC*, pages 238–246. Springer, 1981.

[68] R. Mayr, S. Schewe, and P. Totzke. MDPs with energy-parity objectives. In *LICS '17, IEEE Computer Society*, pages 1–12, 2017.

[69] P. Merlin and D. Faber. Recoverability on communication protocols - implications of a theoretical study. *IEEE Trans. on Communications*, 4(9):1036–1043, 1976.

[70] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[71] A. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *LNCS*, pages 157–168, 1984.

[72] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[73] J. Norris. *Markov Chains*. Cambridge University Press, 1998.

[74] D. Ornstein. On the existence of stationary optimal strategies. In *Proc. American Mathematical Society*, volume 20, pages 563–569, 1969.

[75] C. A. Petri. Kommunikationen mit automaten. *PhD. Thesis, University of Bonn*, 1962.

[76] M. Puterman. *Markov Decision Processes*. Wiley, 1994.

[77] C. Rackoff. The covering and boundedness problems for vector addition systems with states. *Theoretical Computer Science*, 6(2):223–231, 1978.

[78] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. *PhD. Thesis, MIT*, 1973.

[79] J.-F. Raskin, M. Samuelides, and L. V. Begin. Games for counting abstractions. *Electronic Notes in Theoretical Computer Science*, 128(6):69–85, 2005.

[80] W. Reisig. *Petri Nets: An Introduction*. Springer, 1985.

[81] W. Reisig. *A primer in Petri Net design*. Springer-Verlag, 1992.

[82] R. C. Riemann. *Modelling of Concurrent Systems: Structural and Semantical Methods in the High Level Petri Net Calculus*. Herbert Utz Verlag, 1999.

[83] V. V. Ruiz, F. C. Gomez, and D. d. F. Escrig. On non-decidability of reachability for timed-arc Petri nets. In *International Workshop on Petri Nets and Performance Models*, pages 188–196. IEEE Computer Society, 1999.

[84] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, Oct. 1953.

[85] J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 3536 of *Lecture Notes in Computer Science*, pages 385–402. Springer, 2005.

[86] P. Turchin. *Complex Population Dynamics: a Theoretical/Empirical Synthesis*. Princeton University Press, 2003.

[87] M. V. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. of FOCS '85*, pages 327–338, 1985.